# High Precision, High Performance Simulations of Astrophysical Stellar Systems

Mario Spera

*January 2014*

Sapienza - Universita di Roma

SAPIENZA
UNIVERSITÀ DI ROMA

Facolta di Scienze MM. FF. NN.
Dipartimento di Fisica

Tesi di dottorato in Astronomia XXVI ciclo

# High Precision, High Performance Simulations of Astrophysical Stellar Systems

*Candidate* Mario Spera

*Supervisor* Prof. Roberto Capuzzo Dolcetta

January 2014

**Mario Spera**

*High Precision, High Performance Simulations of Astrophysical Stellar Systems*

Tesi di dottorato in Astronomia XXVI ciclo, January 2014

Supervisor: **Prof. Roberto Capuzzo Dolcetta**

**Sapienza - Universita di Roma**

Dipartimento di Fisica

Facolta di Scienze MM. FF. NN.

Piazzale Aldo Moro 5

00185 Roma

# Contents

# Summary

The main target of this work is the discussion of the modern techniques (software and hardware) apt to solve numerically the $N$-body problem in order to develop a numerical code with highest as possible speed and accuracy performance. In particular, we will introduce a new high precision, high performance, code (called `HiGPUs` ) which solves the $N$-body problem exploiting both a high order time integration algorithm (the Hermite's 6th order integrator) and the modern hardware represented by Graphics Processing Units (GPUs), which work as powerful computing accelerators. I will describe in details `HiGPUs` showing how GPUs can be efficiently exploited for gravitational $N$-body simulations up to a large number of particles ($N \simeq 10^7$) with a degree of precision and speed impossible to reach until 5 years ago. Being quite new technologies, the GPUs have not been fully exploited so far; this is why, in this Thesis, I will discuss modern numerical techniques associated with the $N$-body problem, starting from the set up of initial conditions up to the computation of the dynamical evolution of dense and populous stellar systems using GPUs and the two main languages (OpenCL and CUDA) apt to program them.

I will present also results of the application of `HiGPUs` to study the emerging state, and rapid mass segregation, of intermediate-$N$, young, stellar systems after their violent relaxation process. These objects have been investigated simulating systems composed by stars of different

masses, including a central star-mass black hole as well as a model of gas residual of the mother cloud, starting from "cold" to "warm" initial conditions. Moreover, thanks to the high adaptability of the developed software, our group is investigating the formation and the evolution of the innermost region of galaxies (Nuclear Star Clusters). This is, surely, a modern topic, which has not yet received an adequate self-consistent explanation neither from theoretical nor a numerical point of view.

In chapter 1, I will present an historical introduction of the $N$-body problem, describing also its mathematical formulation and the issues which have to be faced when it is considered from a numerical point of view. In chapter 2, I will discuss some modern technologies, in particular GPUs, that can be efficiently used to solve numerically the $N$-body problem while chapter 3 is dedicated to our new code `HiGPUs` . I describe also the tests done on a modern GPU-supercomputer (hosted by the italian supercomputing consortium CINECA) exploiting simultaneously the power of 256 GPUs. I tested the `HiGPUs` performance when running on single, different GPUs in order to understand what of them are the best choice to perform astrophysical $N$-body simulations with convenient balance of accuracy and speed performance. This study is important because it gives a practical contribution to increase the limit of the number of particles that, nowadays, can be simulated using modern hardware and software. In chapter 4, I will face the important problem of generating initial conditions for stellar systems in spherical symmetry describing both the mathematical and numerical tools which stand behind this problem and introducing a numerical utility, which will be soon included in `HiGPUs` , that considers also the presence of a central super massive compact object (for example, a super massive black hole). In chapter 5 I will discuss the main characteristics of a numerical routine, which will be included soon in `HiGPUs` , which implements the so called Mikkola's algorithmic regularization of the $N$-body problem which allows to eliminate the $\propto 1/r$ divergence of the newtonian pair gravitational potential which causes obvious numerical

problems. Finally, as said before, in chapter 6, I will present some (preliminary) results concerning the application of the code `HiGPUs` to study the emerging state of young stellar systems from their violent relaxation, focusing our attention on their resulting degree of mass segregation.

# The $N$-body problem 1

## 1.1 Historical Introduction

The classical gravitational $N$-body problem consists in studying the motion of $N$ point-like masses, placed at points $P_1, P_2, \ldots, P_N$ with velocities $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_N$, interacting through no other forces than their mutual gravitational attraction which is expressed explicitly by the Newton's law (1687, *Philosophiae Naturalis Principia Mathematica*). The development of an appropriate mathematical model for this problem represented a reliable approach for astrophysicists who wanted to give an exhaustive representation of objects from planetary systems up to galaxy clusters. Already in 1710, Johann Bernoulli provided a complete solution for the classical two-body problem although more than 250 years passed before Q. Wang in 1991 [96] got to a convergent power series solution for a generic number of bodies.

### 1.1.1 The King Oscar's Prize

The mathematical and numerical solution of the $N$-body problem has always been considered of primary importance insomuch as between 1885 and 1886 a prize, in honour of King Oscar II of Sweden and Norway, was advertised. For this occasion, Karl Weierstrass formulated the first question which had to be answered no later than the 21st January 1889 (the King's 60th birthday). The problem read:

> *Given a system of arbitrarily many mass points that attract each other according Newton's laws, under the assumption that no two points ever collide, try to find a representation of*

> *the coordinates of each point as a series in a variable that is some known function of time and for all of whose values the series converges uniformly. [...] In the event that the problem remains unsolved at the close of the contest, the prize may also be awarded for a work in which some other problem of Mechanics is treated as indicated and solved completely.*

Unfortunately, none of the 12 papers submitted solved the main problem even if the prize was awarded to Henri Poincaré for his work on Hamiltonian systems.

### 1.1.2 Attempts to find an exact solution

In 1887, the German mathematician Ernst Heinrich Bruns showed that "*the N-body problem has no integrals-algebraic with respect to the time, the position and the velocity coordinates-except the 10 known ones*". Therefore, having ten known integrals of motion, the system of $6N$ equations in $6N$ unknowns is reduced to $6N - 10$ variables. Actually, it is possible to show that the $N$-body problem can be reduced to the order $6N - 12$ by eliminating the explicit dependence on time and by applying the method of the elimination of nodes (see, for example, Boccaletti and Pucacco [19]). The latter strategy is due to Jacobi who applied this technique to the case of $N = 3$ but it can be shown easily that still holds for a generic $N$. It is straightforward that for typical astrophysical values of $N$, the knowledge of just 12 integrals of the motion is not enough to obtain a complete mathematical characterization of the $N$-body problem. Because of these just partial results, the Scientific Community began to believe that the problem was unsolvable and, still nowadays, some wrong echoes tend to resound in the air. Actually, from a purely mathematical point of view, the $N$-body problem is solvable, in fact, it can be shown that a real analytical solution exists (by virtue of the Cauchy's existence theorem), in an open time interval $|t - t_0| < \delta$, provided that the

initial conditions are such that the quantity $\rho = \min_{i \neq j} r_{ij}(t_0)$, where $r_{ij}(t_0)$ is the distance between the $i$ -th and $j$-th particle at time $t_0$, is strictly positive. Indeed, this is a local solution but, it can be analytically extended for $t > t_0 + \delta$ by treating the singularities (collisions between two particles) as elastic bounces. This is the approach followed by Karl Sundman who obtained, in 1913, a series solution in power of $t^{1/3}$, for the three-body problem, uniformly convergent for all real value of $t$. His solution, unfortunately, is not applicable if the initial conditions are such that the system collapses producing a three-body collision (corresponding to an initial angular momentum equal to zero); this because of the inapplicability of the theory of the two-body elastic collision to three bodies.

In 1991, Quidong Wang, a Chinese student, provided, definitively, the solution (in terms of power-series) of the $N$-body problem [96]. His solution, although elegant and remarkable from a mathematical point of view, is not practically relevant. In fact it has a significantly slow convergence and one should sum a very large number of contributions to get to a sufficiently accurate solution of the motion of the particles on a quite short interval of time. This explains why, probably, this theoretically brilliant approach is not widely known and, at the same time, why the main way to solve the $N$-body problem is numerical.

### 1.1.3 The importance of computing facilities

It is already clear that themes apparently a bit far from purely astrophysical studies such as advanced numerical techniques (closer to mathematics) and the development of ever growing computing technologies (closer to computer science) becomes of very big relevance when we talk about the numerical solution of the $N$-body problem or, more in general, about the modern way to do scientific research. In particular, in the last 20 years, computers and supercomputers have led to a deep

transformation of the way to do science yielding to a new model of scientist which is nowadays far to be the character armed with pen and paper alone in his laboratory on his desk. The modern scientist has also a deep technical knowledge on the ways to model, manage, analyse and visualize a scientific problem using the most efficient ways offered by the modern technologies. Today, thanks to this large view and to ever growing equipments, scientific problems related to medicine, physics, climate, energy supplies, global food and water, car crashes, air pollution, and many others, that some decades ago could not be faced because of their considerable complexity, can be now investigated with an unprecedented degree of precision, speed, cheapness and elegance without too much effort. Obviously, the same holds for Astrophysics too. In fact, considering the typical astrophysical scales, formulas and numbers, sooner or later, the ability to easily transform the theoretical model to a numerical one and, therefore, the efficient use of (super)computers in order to solve it, is inevitable. This is something more than pure astrophysics because it requires both a deep knowledge of the scientific problem which has to be faced and the capability to think again of that model using a completely different point of view in order to begin to understand sides which where hidden before when it was written on a piece of paper in the form of complicated and, very often, analytically unsolvable formulas.

### 1.1.4 Past of the numerical $N$-body simulations

The development of advanced numerical techniques associated with the new hardware technologies has been fundamental for the $N$-body problem; in fact, although the mathematical formulation has remained intact since $\sim$XVIII century, we have had to wait the year 1941 to see the first simulation of a self-gravitating $N$-body system. It was carried out by Holmberg [54] even if, at that time, computers and, generally, computational facilities did not exist yet.

**Figure 1.1:** This figure has been taken by the original paper by Holmberg [54]. This figure shows the tidal deformation resulting from the mutual interaction of two galaxies, sampled using 37 light bulbs each, which, in the left panel are assumed in clockwise rotation while in the right panel their rotation is counterclockwise. This is considered the first simulation of a $N$-body system.

Nevertheless, he found a strategy to calculate reciprocal interactions by elegantly replacing gravitation by light. He used light bulbs, which represented the individual mass elements, and measured the total light along two different axes (x and y) by a combination of a photocell and a galvanometer. Since the light obeys an inverse square law, just like gravity, the data collected by Holmberg provided an estimate of the gravitational field and the forces on the individual objects could be evaluated. With this experiment the scientist tried to study the interaction between two massive objects, like galaxies, represented by two circular groups of lamps (A and B), each set with a diameter of 80 cm and each composed by 37 elements (see Fig. 1.1). Holmberg spent weeks in order to set-up and perform this 74-body simulation and the time extension was quite short. For more efficient simulations, we have to wait for the 1960s; in fact, in these years, the first digital computers were

introduced. Although these general purpose machines were very heavy, large, difficult to program and expensive in terms of power consumption, they represented a significant step forward with respect to the Holmberg's experiment yielding to a quicker evaluation of the mutual distances between the $N$ particles. Thanks to the newer technologies, in 1963 a simulation with $N = 100$ was performed by Sverre Aarseth who can be considered, in all respects, the father and the main pioneer of the gravitational $N$-body simulations. Moreover, the simulation brought to an end in 1963 carried out some general results on mass segregation because a mass spectrum for the stars was included. Nevertheless, the first integration methods were very primitive and largely based on trials and errors but every scientist involved in the $N$-body sphere tried to develop something newer, more precise and computationally more efficient improving accuracy and speeding-up the algorithms. In 1985 the number of particles that a computer could evolve for a certain (relatively short) astrophysical time with an acceptable accuracy was $\sim 1000$ (see Aarseth [1]).

Around the end of years 80s, some special purpose machines were developed to increase further the number of particles which could be numerically evolved. Very famous (at least in the field of numerical and theoretical astrophysics) is the so called `GRAvityPipE` (`GRAPE`) project founded by Sugimoto, Hut and Makino. The `GRAPE` boards constitute actual "gravity accelerators" because they are thought to accelerate the evaluation of the mutual distances between all the $N$ particles. They are attached to a host workstation achieving spectacular speedups and they are in use still nowadays (`GRAPE-6`). The main disadvantages is a relatively short mean time between failure, a limited availability and the on-board memory to store data for the $N$ stars.

In the meanwhile, computing machines started to become quite popular so that, today, most of us have a computer (desktop, notebook or recent smartphones too) that can be considered, in all respects, a per-

sonal, always available, powerful computing machine. Initially, *Central Processing Units* (CPUs), which perform the basic arithmetical, logical, and input/output operations on a generic computer, where composed by a single main unit (core) and this was true within 2005. Before 2005 the strategy to improve performance was, in broad lines, increasing both the speed (clock frequency) of the single core and the cache memory (small in size but very fast in terms of access). After 2005, because of the even growing complexity of the operations (also scientific) to accomplish, the idea of "parallelism" gained ground. The clock frequency of the core could be significantly reduced, improving power consumption, because, thanks to this approach, many calculations could be carried out simultaneously dividing a large problem into smaller ones. This was a very important step for $N$-body simulations because the step which evaluates the mutual forces, being the force between particles $i$ and $j$ completely independent from that between particles $k$ and $s$ with $i \neq j \neq k \neq s$, can be performed in parallel. If we suppose to have $\eta$ cores on a single workstation, one very simple approach is to distribute particles in equal parts over the cores in order to calculate forces simultaneously. In this way the computing time, theoretically, decreases by a factor $\eta$ with respect to that spent using just one core. Nowadays it is very common to find on the market CPUs specifically dedicated to High Performance Computing (HPC) composed by 6 or 8 physical cores (12 or 16 virtual) and almost every notebook or desktop pc harbours, at least, a quad-core CPU. However, when we deals with a system with a number of stars close to the astrophysical reality, $N \gtrsim 10^5$, exploiting a multi-core CPU could not be enough to get scientific results in a reasonably short computing time. All the $N$-body simulations, until $\sim$2006, were made using CPUs or special-purpose machines. In the last 5-10 years, a great help, in this sense, came from the gaming industry (even if not specifically for scientific research but, rather, for performing specific rendering applications to boost the frame-rate of video-games): *Graphics Processing Units* (GPUs). Graphic devices are slowly replacing CPUs and dedicated hardware for a series of numerical applications

because they are getting cheaper and faster while keeping the electric power consumption at very low levels. A GPU is particularly suitable for parallel computing because, in a video-frame, many pixels should be updated at the same time at fast rates and, at the same time, each pixel does not require information from other pixels. The growth of GPUs as means for scientific computing is strictly linked with the introduction of *Compute Unified Device Architecture* (CUDA, 2006), introduced by the nVIDIA corporation; in fact, thanks to this novelty, nVIDIA graphic cards became easily programmable. CUDA (now at version 5.5) is of simple use because is based on the C programming language even if its limitation is that it can be used to exploit GPUs of the nVIDIA make only. Recently, another GPU programming language has been introduced by the Khronos group: *Open Computing Language* (OpenCL, 2008). OpenCL is based on the programming language C99 and can be used to manage GPUs of different vendors (nVIDIA, AMD, etc...) as well as CPUs.

## 1.2 Mathematics of the $N$-body problem

### 1.2.1 Newton's law and equations of motion

We start our mathematical discussion considering a particle of mass $M$, placed at the origin of a Cartesian coordinate system, and a vector $\mathbf{r}$ which gives the position of a particle of mass $m_p$, placed at a generic point $P(x, y, z)$. The force acting on this particle, according to Newton's law, is

$$\mathbf{F} = -G\frac{Mm_p}{r^2}\mathbf{e_r} \tag{1.1}$$

where $G$ is the universal gravitational constant, $\mathbf{e_r} = \frac{\mathbf{r}}{r}$ is the radial unit vector, which indicates that the force is directed along the line joining the two particles, and the minus sign denotes an attractive force. We

define the single-particle potential $U(x, y, z)$ as the scalar function such that

$$\nabla U = \mathbf{F} \tag{1.2}$$

where

$$\nabla \equiv \left( \frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z} \right) \ . \tag{1.3}$$

Given that it is possible to write

$$\nabla G\frac{Mm_p}{r} = -G\frac{Mm_p}{r^2}\nabla r = -G\frac{Mm_p}{r^2}\frac{\mathbf{r}}{r} = -G\frac{Mm_p}{r^2}\mathbf{e_r} \tag{1.4}$$

we obtain

$$U(x, y, z) = U(r) = G\frac{Mm_p}{r} \ . \tag{1.5}$$

Introducing the independent variable $t$, which represents time, the equation of the motion of the individual particle writes

$$m_p\ddot{\mathbf{r}} = -G\frac{Mm_p}{r^3}\mathbf{r} \tag{1.6}$$

which, completed with a set of initial conditions, leads to

$$\begin{cases} \ddot{\mathbf{r}} & = -G\frac{M}{r^3}\mathbf{r} \\ \mathbf{r}(t_0) & = \mathbf{r}_0 \\ \dot{\mathbf{r}}(t_0) & = \dot{\mathbf{r}}_0 \end{cases} \ . \tag{1.7}$$

This constitutes a set of differential equations which represents a Cauchy's problem of one second-order vector equation or, equivalently, of three second-order scalar equations which can be reduced to a system of six first-order, scalar equations putting $\dot{\mathbf{r}} = \mathbf{v}$ and $\mathbf{v} = -G\frac{m}{r^3}\mathbf{r}$. The case of $N$ bodies is an immediate generalization of the two-body treatment. The resulting force $\mathbf{F}_i$, acting on the $i$-th particle, is the sum of the forces $\mathbf{F}_{ij}$ due to the attraction of all the other $N-1$ bodies. Therefore

$$\mathbf{F}_i = \sum_{\substack{j=1 \\ j\neq i}}^{N} G\frac{m_i m_j}{r_{ij}^3}\left(\mathbf{r}_j - \mathbf{r}_i\right) \tag{1.8}$$

where we have introduced

$$r_{ij} = |\mathbf{r}_j - \mathbf{r}_i| = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2 + (z_j - z_i)^2} \tag{1.9}$$

which is the module of the distance between particle $i$ and particle $j$. First of all, it is worth noting the symmetry of the gravitational force; in fact, from (1.8), we have

$$\mathbf{F}_{ij} = -\mathbf{F}_{ji} \tag{1.10}$$

which is an important property that we will use widely to show some theoretical peculiarities of the $N$-body problem. It can be shown that the generalization of the potential function $U$ is

$$U(\mathbf{r}_1, \mathbf{r}_2, ..., \mathbf{r}_N) \equiv \frac{1}{2} \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j \neq i}}^{N} G \frac{m_i m_j}{r_{ij}} \tag{1.11}$$

being trivial to prove that $\nabla_k U = \mathbf{F}_k$. At this point it is possible to generalize system 1.7 in order to obtain the mathematical representation of the classical gravitational $N$-body problem which is characterized by a system of $N$ second-order differential equations

$$\begin{cases} \ddot{\mathbf{r}}_i & = \sum_{\substack{j=1 \\ j \neq i}}^{N} G \frac{m_j}{r_{ij}^3} (\mathbf{r}_j - \mathbf{r}_i) \\ \mathbf{r}_i(t_0) & = \mathbf{r}_{i0} \\ \dot{\mathbf{r}}_i(t_0) & = \dot{\mathbf{r}}_{i0} \end{cases} . \tag{1.12}$$

System 1.12 is reducible, like the two-body case, to a system of $6N$ first-order scalar equations putting $\dot{\mathbf{r}}_i = \mathbf{v}_i$ and $\mathbf{v}_i = \frac{1}{m_i} \frac{\partial U}{\partial \mathbf{r}_i}$. Although the theoretical formulation of the problem is very simple, its numerical resolution, as we will discuss later on, presents significant difficulties.

## 1.2.2 The integrals of the motion

First of all, it is worth formulating a definition of integral of motion. Referring, specifically, to an $N$-body system, an *integral of motion* is any function $I$ of the phase-space coordinates of the $N$ stars only, $(\mathbf{x}, \mathbf{v})$, which is constant along the motion of a generic particle, that is

$$I\left[\mathbf{x}(t_1), \mathbf{v}(t_1)\right] = I\left[\mathbf{x}(t_2), \mathbf{v}(t_2)\right] \Rightarrow \frac{\mathrm{d}I\left[\mathbf{x}\left(t\right), \mathbf{v}\left(t\right)\right]}{\mathrm{d}t} = 0 \ . \tag{1.13}$$

The integrals which are able to confine orbits in the phase-space are called *isolating integrals* and, it can be shown that, for an $N$-body system, up to 12 isolating integrals can be found; specifically, here we focus our attention on ten of them which can be determined using the Newtonian formalism while the remaining 2 can be made explicit using other more sophisticated procedures (for a detailed discussion about this topic see, for example, [19]).

### Total energy conservation

Let us start examining the total energy of the system $E$, which is

$$E = T - U = \frac{1}{2}\sum_{i=1}^{N} m_i \dot{r}_i^2 - \frac{1}{2}\sum_{i=1}^{N}\sum_{\substack{j=1 \\ j\neq i}}^{N} G\frac{m_i m_j}{r_{ij}} \tag{1.14}$$

where $T$ is the total kinetic energy and $U$ has already been defined in (1.11). It can be shown that the energy $E$ is an integral of the motion by demonstrating the validity of the relation

$$\dot{E} = \dot{T} - \dot{U} = 0 \ . \tag{1.15}$$

In fact, we know that

$$\dot{T} = \frac{1}{2}\sum_{i=1}^{N} 2m_i \dot{\mathbf{r}}_i \cdot \ddot{\mathbf{r}}_i = \sum_{i=1}^{N} m_i \dot{\mathbf{r}}_i \cdot \ddot{\mathbf{r}}_i \tag{1.16}$$

and, since the potential function depends on time through the positions, we may write

$$\dot{U}(x_1, y_1, z_1, ..., x_N, y_N, z_N) = \sum_{i=1}^{N} \left( \frac{\partial U}{\partial x_i}\frac{\partial x_i}{\partial t} + \frac{\partial U}{\partial y_i}\frac{\partial y_i}{\partial t} + \frac{\partial U}{\partial z_i}\frac{\partial z_i}{\partial t} \right) = \sum_{i=1}^{N} \nabla_i U \cdot \dot{\mathbf{r}}_i \ . \tag{1.17}$$

Having $\mathbf{F}_i = m_i \ddot{\mathbf{r}}_i = \nabla_i U$, substituting into (1.17) we obtain

$$\dot{U} = \sum_{i=1}^{N} m_i \dot{\mathbf{r}}_i \cdot \ddot{\mathbf{r}}_i = \dot{T} \tag{1.18}$$

therefore, we have proven that the total energy is one of the integrals of the motion letting the order of the system (1.12) be reduced to $6N-1$.

## Total angular momentum conservation

The total angular momentum of an $N$-body system is given by

$$\mathbf{L} = \sum_{i=1}^{N} m_i \mathbf{r}_i \wedge \dot{\mathbf{r}}_i \tag{1.19}$$

so, we will show that, for an isolating system, $\dot{\mathbf{L}} = \mathbf{0}$ and, consequently, the total angular momentum is a further integral of motion. To prove it, first of all, we need to recover the second cardinal equation of dynamics. It states that the time derivative of the total angular momentum of a generic system is equal to the sum of the resulting moment of the external ($\mathbf{M}^{(\text{ext})}$) and internal ($\mathbf{M}^{(\text{int})}$) forces, i.e.

$$\dot{\mathbf{L}} = \mathbf{M}^{(\text{ext})} + \mathbf{M}^{(\text{int})} = \sum_{i=1}^{N} \mathbf{r}_i \wedge \mathbf{F}_i^{(\text{ext})} + \sum_{i=1}^{N} \mathbf{r}_i \wedge \mathbf{F}_i^{(\text{int})} \ . \tag{1.20}$$

In our case, if we consider an isolated $N$-body system, $\mathbf{M}^{(\mathrm{ext})} = \mathbf{0}$ therefore we have

$$\dot{\mathbf{L}} = \sum_{i=1}^{N} \mathbf{r}_i \wedge \mathbf{F}_i \ . \tag{1.21}$$

Thanks to the relation (1.8), we can replace the quantity $\mathbf{F}_i$ in the expression (1.21) obtaining

$$\dot{\mathbf{L}} = \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j \neq i}}^{N} \mathbf{r}_i \wedge \mathbf{F}_{ij} \ . \tag{1.22}$$

The summation (1.22) contains both the terms $\mathbf{r}_i \wedge \mathbf{F}_{ij}$ and $\mathbf{r}_j \wedge \mathbf{F}_{ji}$ so, instead of writing a double sum, respectively on the index $i$ and $j$, we can write

$$\dot{\mathbf{L}} = \sum_{\substack{(i,j)=1 \\ j \neq i}}^{N} (\mathbf{r}_i \wedge \mathbf{F}_{ij} + \mathbf{r}_j \wedge \mathbf{F}_{ji}) \tag{1.23}$$

which represents a single summation on the couple of values $(i, j)$. Specifically, the formula (1.23) contains only $\binom{N}{2} = \frac{N(N-1)}{2}$ terms of type $(\mathbf{r}_i \wedge \mathbf{F}_{ij} + \mathbf{r}_j \wedge \mathbf{F}_{ji})$. Taking into account the relation (1.10), we have

$$\dot{\mathbf{L}} = \sum_{\substack{(i,j)=1 \\ j \neq i}}^{N} (\mathbf{r}_i \wedge \mathbf{F}_{ij} - \mathbf{r}_j \wedge \mathbf{F}_{ij}) = \sum_{\substack{(i,j)=1 \\ j \neq i}}^{N} G \frac{m_i m_j}{r_{ij}^3} (\mathbf{r}_i - \mathbf{r}_j) \wedge (\mathbf{r}_j - \mathbf{r}_i) = \mathbf{0} \tag{1.24}$$

where, in the last passage, we have used the explicit expression of $\mathbf{F}_{ij}$ given by (1.8). From the expression (1.24) follows that $\mathbf{L}$ is an integral of motion, thing that allows to reduce the order of the system (1.12) to $6N - 4$.

## Centre of mass position and velocity

To get to the explicit expression of another integral, let us consider the equation of the motion

$$m_i \ddot{\mathbf{r}}_i = \sum_{\substack{j=1 \\ j \neq i}}^{N} G \frac{m_i m_j}{r_{ij}^3} (\mathbf{r}_j - \mathbf{r}_i) \ ; \tag{1.25}$$

summing both sides over $i$, we obtain

$$\sum_{i=1}^{N} m_i \ddot{\mathbf{r}}_i = \mathbf{0} = \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j \neq i}}^{N} G \frac{m_i m_j}{r_{ij}^3} (\mathbf{r}_j - \mathbf{r}_i) = \mathbf{0} \tag{1.26}$$

because we have obtained a sum of terms $\mathbf{r}_j - \mathbf{r}_i$ and $\mathbf{r}_i - \mathbf{r}_j$ which have opposite signs and cancel each other two by two. The relation (1.26) can be expressed in a more useful form introducing the position of the centre of mass of the system

$$\mathbf{r}_{c.m.} = \frac{\sum_{i=1}^{N} m_i \mathbf{r}_i}{\sum_{i=1}^{N} m_i} = \frac{1}{M} \sum_{i=1}^{N} m_i \mathbf{r}_i \tag{1.27}$$

where $M$ is the total mass of the system. Therefore, the relation (1.26) is equivalent to

$$\ddot{\mathbf{r}}_{c.m.} = \mathbf{0} \tag{1.28}$$

that is

$$\dot{\mathbf{r}}_{c.m.} = \frac{1}{M} \sum_{i=1}^{N} m_i \dot{\mathbf{r}}_i = \frac{\mathbf{Q}}{\mathrm{M}} = \text{constant} \tag{1.29}$$

which express that the total momentum $\mathbf{Q}$ is an integral of motion and it allows us to reduce the order of the system (1.12) of 3 units (one per component). If we choose a new system of reference such that the initial velocity of the centre of mass is null we have

$$\dot{\mathbf{r}}_{c.m.} = \frac{\mathbf{Q}}{M} = \mathbf{0} \tag{1.30}$$

and, integrating again, we obtain another integral of motion such that

$$\mathbf{r}_{c.m.} = \text{constant} = \mathbf{0} \tag{1.31}$$

which, provided to put the origin of the system of reference in the centre of mass, definitively, reduces the order of the main system to $6N - 10$.

### 1.2.3 The virial theorem

Although Clausius in 1870 formulated the virial theorem to study the mechanical origin of heat [33], his theory, very soon, was adapted to other problems, including stellar dynamics and, specifically, the $N$-body problem. To derive the compact expression of this theorem, we may start from the equation of the motion of a generic particle $k$, belonging to an $N$-body system, written in terms of the derivative of the potential function

$$m_k \ddot{\mathbf{r}}_k = \frac{\partial U}{\partial \mathbf{r}_k} \ . \tag{1.32}$$

Taking into account that

$$\frac{1}{2}\frac{\mathrm{d}^2}{\mathrm{d}t^2}\left(\mathbf{r}_k \cdot \mathbf{r}_k\right) = \frac{\mathrm{d}}{\mathrm{d}t}\left(\mathbf{r}_k \cdot \dot{\mathbf{r}}_k\right) = \left|\dot{\mathbf{r}}_k\right|^2 + \mathbf{r}_k \cdot \ddot{\mathbf{r}}_k \tag{1.33}$$

and multiplying (1.32) by $\mathbf{r}_k$, we get

$$\frac{1}{2}\frac{\mathrm{d}^2}{\mathrm{d}t^2}\left(m_k r_k^2\right) = m_k \left|\dot{\mathbf{r}}_k\right|^2 + \mathbf{r}_k \cdot \frac{\partial U}{\partial \mathbf{r}_k} \ . \tag{1.34}$$

Summing over $k$ and multiplying by $\frac{1}{2}$, we have

$$\frac{1}{4}\frac{\mathrm{d}^2}{\mathrm{d}t^2}\sum_{k=1}^{N} m_k r_k^2 = \sum_{k=1}^{N}\frac{1}{2} m_k \left|\dot{\mathbf{r}}_k\right|^2 + \frac{1}{2}\sum_{k=1}^{N} \mathbf{r}_k \cdot \frac{\partial U}{\partial \mathbf{r}_k} \ . \tag{1.35}$$

If we introduce the *polar moment of inertia* of the system

$$I = \sum_{k=1}^{N} m_k r_k^2 \tag{1.36}$$

and, if we remember that the first term on the right side of equation (1.35) is the total kinetic energy of the system $T$, we can express the relation (1.35) in a more elegant form

$$\frac{1}{4}\frac{\mathrm{d}^2 I}{\mathrm{d}t^2} = T + \frac{1}{2}\sum_{k=1}^{N} \mathbf{r}_k \cdot \frac{\partial U}{\partial \mathbf{r}_k} \ . \tag{1.37}$$

This is not the final form of the virial theorem yet because it is possible to express the so called *Clausius' virial* (the last term on the right side of equation (1.37) ) in a more convenient form. To do this, it is worth noting that a real function $f(\mathbf{r})$, of $m$ real variables, is said to be *homogeneous* of degree $n$ if

$$f(\alpha \mathbf{r}) = \alpha^n f(\mathbf{r}) \tag{1.38}$$

$\forall \alpha \in \Re$ ($\alpha \neq 0$) and $\forall \mathbf{r} \in \Re^m$. From (1.38) it can be shown that the potential function $U(\mathbf{r}_1, \mathbf{r}_2, ..., \mathbf{r}_N)$ is homogeneous of degree $n = -1$. At the light of this property, we can write

$$\frac{\partial U(\alpha \mathbf{r})}{\partial \alpha} = \frac{\partial (\alpha^n U)}{\partial \alpha} = n\alpha^{n-1} U \ . \tag{1.39}$$

Nevertheless we have

$$\frac{\partial U(\alpha \mathbf{r})}{\partial \alpha} = \sum_{i=1}^{N} (\nabla_{\alpha \mathbf{r}_i} U) \cdot \mathbf{r}_i \ . \tag{1.40}$$

Since $\alpha$ is arbitrary, we choose $\alpha = 1$ and, using both the equalities (1.39) and (1.40), we obtain

$$\sum_{i=1}^{N} \nabla_{\mathbf{r}_i} U \cdot \mathbf{r}_i = -U \ . \tag{1.41}$$

Substituting (1.41) in (1.35) we get

$$\frac{1}{2}\ddot{I} = 2T - U \tag{1.42}$$

which represents the final form of the virial theorem. Sometimes, the potential energy $\Omega = -U$ is introduced and the equation (1.42) takes the form

$$2T + \Omega = \frac{1}{2}\ddot{I} \tag{1.43}$$

which can also be expressed, taking into account that $E = T - U = T + \Omega$, in the following way

$$E + T = \frac{1}{2}\ddot{I} \ . \tag{1.44}$$

### Consequences of the virial theorem

Generally, an $N$-body system is said to be *stable*, in the sense that it remains confined to a limited region of the space, if the following conditions are verified :

1. $r_{ij}(t) \neq 0$ for every $i \neq j$ at any $t$;

2. $|r_{ij}(t)| < A$ for any $t$ , where $A$ is a positive constant.

A necessary condition for this to happen is that $E < 0$. To demonstrate this, we can start from (1.44) because, if $E > 0$, we may write

$$\frac{1}{2}\ddot{I} \geq E \tag{1.45}$$

and, integrating two times, we have

$$I(t) \geq Et^2 + \dot{I}(t_0)t + I(t_0) \tag{1.46}$$

which grows quadratically in $t$ yielding to $I \to \infty$ when $t \to \infty$, so, from (1.36), $r_k \to \infty$ at least for one value of $k$ and this does not verify the condition listed above with the number 2. Therefore $E < 0$ is a necessary, but not sufficient, condition; in fact, in an $N$-body system, the energy per particle does not represent a conserved quantity, therefore, although $E < 0$ is verified, a particle in position $\mathbf{r}_p$ , locally, can reach (and maintain) an energy $E_p$ such that

$$E_p = \frac{1}{2}m_p v_p^2(\mathbf{r}_p) - U(\mathbf{r}_p) > 0 \tag{1.47}$$

that is

$$v_p(\mathbf{r}_p) > \sqrt{\frac{2U(\mathbf{r}_p)}{m_p}} = v_e^{(\mathrm{p})}(\mathbf{r}_p) \tag{1.48}$$

where $v_e^{(\mathrm{p})}(\mathbf{r}_p)$ indicates the *escape velocity* for the particle $p$ at position $\mathbf{r}_p$. In this case, the particle $p$ can escape from the system and the condition number 2 is no longer verified.

The virial theorem can give us some information about the global behaviour of the system. To show it, we now introduce the idea of *time averaging*. Given a quantity $A(t)$, its average over the time interval $(0, t)$ is given by

$$\langle A \rangle_t = \frac{1}{t} \int_0^t A(\tau) d\tau \ . \tag{1.49}$$

Now, if we apply the definition (1.49) to (1.42), averaging over a time $t$, we obtain

$$\frac{\dot{I}(t) - \dot{I}(0)}{2t} = 2 \langle T \rangle_t - \langle U \rangle_t \tag{1.50}$$

and, in addition, if the system is limited in the phase space, we can affirm that $\dot{I}(t) - \dot{I}(0)$ is a limited quantity and, if $t \to \infty$, we have

$$2 \langle T \rangle_\infty - \langle U \rangle_\infty = \lim_{t \to \infty} \frac{\dot{I}(t) - \dot{I}(0)}{2t} = 0 \ . \tag{1.51}$$

Therefore, a limited system, after a long time, is said to be *virialized* if it has an average *virial ratio* $\langle Q \rangle_\infty$ such that

$$\langle Q \rangle_\infty = \frac{2 \langle T \rangle_\infty}{\langle U \rangle_\infty} = 1 \ . \tag{1.52}$$

Generally, when $\ddot{I} > 0$ it is clear that $T > -E$ and the system, globally, tends to expand. On the contrary, when $\ddot{I} < 0$ we have $T < -E$ and the system tends to contract. However, even if a system is initially characterized by a value $\ddot{I} \neq 0$, it tends to a virialized condition, which corresponds to gravitational equilibrium, on a time comparable to the relaxation time (see section 1.2.4). This stationary state is reached af-

ter a sequence of expansions and contractions which tend to fade with time and that may well be seen by drawing a plot showing the trend of the virial ratio in time for a generic simulation of a $N$-body system starting from $Q \neq 1$. In this plot, after some relaxation times, it can be noted that the virial ratio fluctuates around the value 1 with statistical fluctuations of the order of $1/\sqrt{N}$. An example of the typical trend of the virial ratio for a generic $N$-body system is shown in Fig, 1.2. There

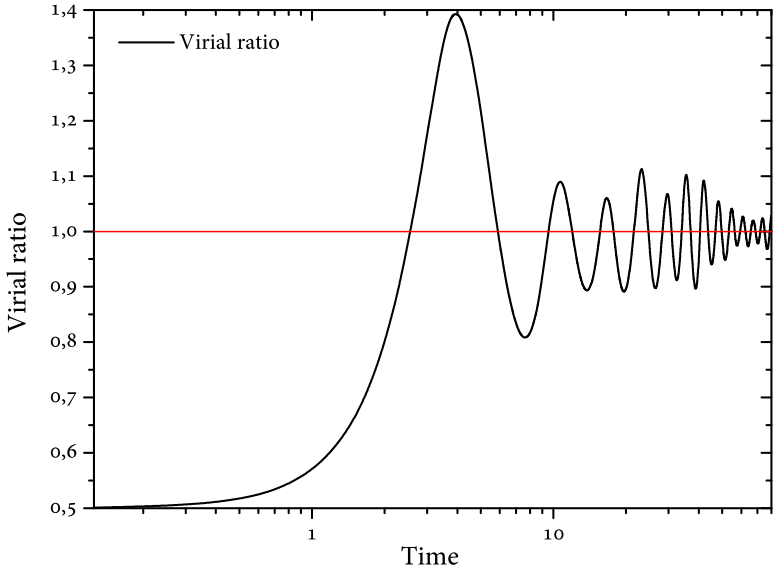


**Figure 1.2:** This figure shows the trend of the virial ratio, in function of time, for a $N$-body system composed by $\sim$ 32,000 particles starting from an initial virial ratio equal to 0.5. It is evident the decreasing amplitude of the oscillations of the virial ratio, around the equilibrium value $Q = 1.0$, as the system evolves in time.

are some astrophysical cases in which the virialization of the system is not verified. This, for example, is what we observed performing some simulations of violent collapses starting from $N$-body systems whose particles were initially posed on rest, $Q = 0$ (see Chapter 6). This is mainly due to the presence of a significant percentage of high velocity

stars that escape from the system just after the violent collapse. This, indeed, violates the condition for which the system must have a limited phase-space. In fact, it is not possible to choose a positive constant $A$ such that $|r_{ij}(t)| < A$ for any $t$ because the members who escape from the $N$-body system always increase their distance from a fixed point in the space, getting to infinity for $t \to \infty$ maintaining approximatively a constant velocity. Therefore, in such cases, $Q \gtrsim 1$ and the larger the percentage of escape stars, the larger the deviation from 1 of the virial ratio. The explicit expression of the virial theorem is a powerful tool which, often, is used to argue some intrinsic characteristics of astrophysical systems from observable quantities. To show this, it is convenient to write down the general expression of the total gravitational potential energy which is

$$U = \int \mathrm{d}^3\mathbf{x}\rho\left(\mathbf{x}\right) \cdot \nabla\phi\left(\mathbf{x}\right) \ , \tag{1.53}$$

where $\rho\left(\mathbf{x}\right)$ is the mass density profile of the system and $\phi\left(\mathbf{x}\right)$ is the gravitational potential. In the case of spherically symmetric system, equation (1.53) can be simplified in

$$U = 4\pi G \int_0^\infty \mathrm{d}r r\rho\left(r\right) M\left(r\right) \ . \tag{1.54}$$

In general, for a generic $N$-body system, equation (1.54) is written in another, simpler form, which is

$$U = \alpha \frac{GM^2}{R} \tag{1.55}$$

where $\alpha$ is the so called *form factor* that, indeed, takes into account the shape of the density profile of the system while $R$ is the characteristic dimension of the stellar system and $M$ its total mass. For example, for a uniform density distribution $\alpha_u = \frac{3}{5}$, for a Plummer model [82] $\alpha_u = \frac{3}{32}\pi$. Moreover, the total kinetic energy of the system may be written as

$$T = \frac{1}{2} M \overline{v^2} \tag{1.56}$$

where

$$\overline{v^2} = \frac{\sum_{i=1}^{N} m_i \left|\mathbf{v}_i\right|^2}{\sum_{i=1}^{N} m_i} \ . \tag{1.57}$$

If a certain stellar system is stationary and consequently virialized, the virial theorem in equation (1.51) may be therefore written again in the form

$$M\overline{v^2} - \alpha\frac{GM^2}{R} = 0 \ . \tag{1.58}$$

Relation (1.58) can be used to determine the so called virial mass of a stellar system starting from the observational parameters $\overline{v^2}$ and $R$. This way is useful to easily determine, for example, mass-luminosity ratios of astrophysical systems, black holes masses or to highlight the presence of dark matter.

### 1.2.4 Typical time scales of an N-body system

The *relaxation time* ($t_c$) is one of the most important parameters used to describe exhaustively the evolution of a generic stellar system. It is defined as the time over which, as a result of collisions between particles, a stellar system completely loses memory of its initial state. After this time, a system is generally said to be *relaxed*. To find an approximated expression of the relaxation time, we consider a test particle, with mass $m$, launched against a system composed by $N$ particles of mass $M$ (field stars), from position $\tilde{\mathbf{r}} \to \infty$ with velocity $\mathbf{v}$ along the x-axis. As we can

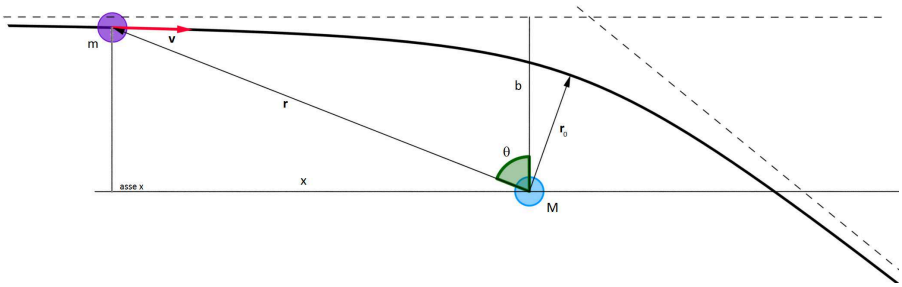


**Figure 1.3:** A simple scheme of a two-body collision with impact parameter $b$.

see in figure (1.3), we assume that the *impact parameter*, related to the generic field star, is $b$. Each close encounter produces a perturbation $\delta \mathbf{v}_\perp$, directed along the y-axis, to the test particle's velocity, but the mean value $\langle \Delta \mathbf{v}_\perp \rangle$, summed over all the encounters, is zero because the field stars are assumed to be distributed uniformly, therefore, to get to an explicit expression of the relaxation time we will impose the condition

$$\frac{\langle \Delta v_\perp^2 \rangle}{v^2} \simeq 1 \tag{1.59}$$

which is equivalent to require that the generic test particle completely loses memory of its initial trajectory. This simplified view will lead us to an approach which is valid only under certain hypothesis which will be clarified later in this thesis. Nevertheless, it is worth following this kind of treatment because it allow us to obtain a very simple formula which, anyway, maintains its validity for a significantly large number of stellar systems. To obtain a relation for $\langle \Delta v_\perp^2 \rangle$, we consider first the single collision represented in figure (1.3); the module of the perpendicular force $F_\perp$, (i.e. the component along the y-axis), acting on the mass $m$, can be written as

$$F_\perp = G\frac{mM}{r^2}\cos\theta \ . \tag{1.60}$$

If we assume that the perturbation to the velocity is small (i.e. $\delta v_\perp / v \ll 1$, therefore we are excluding very close encounters between stars) we can write

$$\cos\theta \simeq \frac{b}{r} \qquad r^2 \simeq x^2 + b^2 \qquad x = vt \tag{1.61}$$

and, substituting into (1.60), we get

$$F_\perp = G\frac{mMb}{(v^2t^2 + b^2)^{\frac{3}{2}}} = G\frac{mM}{b^2\left[\left(\frac{vt}{b}\right)^2 + 1\right]^{\frac{3}{2}}} = m\frac{\mathrm{d}}{\mathrm{d}t}v_\perp \tag{1.62}$$

where, in the last passage, we have used the Newton's second law of motion. If we integrate the relation (1.62) with respect to time, we have

$$\delta v_\perp = \frac{GM}{b^2} \int_{-\infty}^{+\infty} \left[ \left( \frac{vt}{b} \right)^2 + 1 \right]^{-\frac{3}{2}} \mathrm{d}t = 2\frac{GM}{bv} \tag{1.63}$$

where the integral can be solved by putting $\frac{vt}{b} = \sinh x$ and noting that $\mathrm{d}\tanh x = (\cosh x)^{-2}\,\mathrm{d}x$. The average number of collisions $\delta n_b$, with an impact parameter between $b$ and $b + \delta b$, suffered by a particle crossing a system with a typical dimension equal to $R$ is

$$\delta n_b = P(b, R) \cdot N \tag{1.64}$$

where $P(b, R)$ is the probability of the single close encounter which is equal to the ratio between the geometric cross section of the collision and the geometric cross section of the system. Therefore

$$\delta n_b = \frac{2\pi b \mathrm{d}b}{\pi R^2} N = \frac{2bN}{R^2} \mathrm{d}b \tag{1.65}$$

and the mean quadratic variation of $v_\perp$, due to the collisions $\delta n_b$, can be expressed as

$$\left\langle \Delta v_\perp^2 \right\rangle_b = \delta n_b \delta v_\perp^2 = \frac{8G^2M^2N}{v^2R^2} \mathrm{d}\log b \ . \tag{1.66}$$

To evaluate $\langle \Delta v_\perp^2 \rangle$ we need to integrate the relation (1.66) over all possible values of $b$. Considering that the gravitational force never vanishes, it sounds reasonable to choose the typical dimension of the system, that is $R$, as the maximum value of the impact parameter ($b_M$). The minimum value could be $b_m = 0$ but, in this case, integrating (1.66), we should face a logarithmic divergence. Therefore, to get to an estimation of $b_m$, we use the distance of minimum approach between the two particle involved in the collision, that is $r_0$ (see figure (1.3)). To evaluate this quantity, we can use the conservation of the total system (2 bodies) energy

$$\frac{1}{2}\mu v^2 = \frac{1}{2}\mu v_0^2 - G\frac{mM}{r_0} \tag{1.67}$$

where $v_0$ is the velocity at the minimum distance, and $\mu = \frac{mM}{m+M}$ is the reduced mass of the system. From (1.67) we obtain

$$\frac{1}{2}\mu v_0^2 - G\frac{mM}{r_0} > 0 \Rightarrow r_0 > \frac{2G(M+m)}{v^2\frac{v_0^2}{v^2}} \simeq \frac{2G(M+m)}{v^2} = b_m \quad (1.68)$$

where we have used the approximation of small perturbation ($\frac{v_0^2}{v^2} \simeq 1$). At the light of this, we can integrate the relation(1.66) between $b_m$ and $b_M$ obtaining

$$\left\langle \Delta v_\perp^2 \right\rangle = \frac{8G^2M^2N}{v^2R^2}\log\Lambda \quad (1.69)$$

where the quantity $\log\Lambda = \log\frac{b_M}{b_m}$ has been introduced and, often, it is called *Coulomb logarithm*. Dividing (1.69) by $v^2$ we get

$$\frac{\left\langle \Delta v_\perp^2 \right\rangle}{v^2} = \frac{8G^2M^2N}{v^4R^2}\log\Lambda \ . \quad (1.70)$$

From (1.42) we can evaluate the typical velocity of a particle in an $N$-body system, provided that this system is in a stationary (virialized) state. We have

$$2T - U = M_{tot}v_{typical}^2 - \alpha\frac{GM_{tot}^2}{R} = 0 \Rightarrow v_{typical}^2 = \alpha\frac{GM_{tot}}{R} = \alpha\frac{GMN}{R} \quad (1.71)$$

where $\alpha$ is the already introduced form factor (see 1.2.3) and $M_{tot} = MN$ is the total mass of the system. Putting $\alpha = 1$ and substituting into (1.70), we obtain

$$\frac{\left\langle \Delta v_\perp^2 \right\rangle}{v^2} = \frac{8}{N}\log\Lambda \ . \quad (1.72)$$

We now define another important time-scale for a $N$-body system: the *crossing time* $t_c$. This is defined as the time that a particle takes to cross the typical dimension of the system to which it belongs. Therefore

$$t_c = \frac{R}{v} = \frac{R^{\frac{3}{2}}}{\sqrt{GMN}} \ . \quad (1.73)$$

Before the system relaxes, a particle will pass through the system a number of times $n_R$ which can be derived from the condition

$$\frac{\langle \Delta v_\perp^2 \rangle}{v^2} n_R \simeq 1 \Rightarrow n_R \simeq \frac{1}{8} \frac{N}{\log \Lambda} \tag{1.74}$$

so, definitively, we obtain

$$t_r \simeq n_R t_c = \frac{1}{8} \frac{N}{\log \Lambda} t_c \tag{1.75}$$

where, generally, $\log \Lambda$ is replaced by $\log N$ because

$$\log \Lambda = \log \frac{R v^2}{2G(M+m)} \simeq \log \frac{N}{2} \simeq \log N \tag{1.76}$$

where we have used the expression of the typical velocity (1.71) and we have considered $M \gg m$ and $\log N \gg \log 2$. According to formula (1.75), a typical, virialized globular cluster, having a characteristic dimension of about $10$ pc and velocity $10$ km/s, has a crossing time $t_c \simeq 1\text{Myr}$. Assuming that, on average, a typical globular cluster is composed by $\sim 5 \times 10^5$ stars, its relaxation time is about $10$ Gyr which is approximatively the age of such astrophysical system. This means that systems like globular cluster (but also open clusters) are dynamically old. The immediate consequence is that, on average, all the systems belonging to the latter category, appear to eyes almost identical. On the contrary, it is possible to show that galaxies and galaxy clusters are dynamically young systems having a relaxation time of about $10^{13}$ years.

### A more general form of the relaxation time

It is clear from the treatment followed in the previous paragraph that the formula 1.75 relies on the assumption of virial equilibrium of the system. an extended and exhaustive derivation of the relaxation time can be found in [18]. Summarizing it in broad lines, a more general

formulation can be obtained considering the so called *master equation* which is the formula that expresses the evolution, in time, of the distribution function of a generic stellar system when close encounters between stars are taken into account. Using the *Fokker-Planck approximation* the master equation becomes expressible in terms of the so called *diffusion coefficients* which are quantities that denote the expectation of change of a specific phase-space coordinate ($w_i$) per unit of time. Taking into account the diffusion coefficient indicated as $D\left[\left(\Delta v_{\parallel}^2\right)\right]$, the relaxation time is defined as

$$t_{rel} \equiv \frac{v^2}{D\left[\left(\Delta v_{\parallel}^2\right)\right]} \tag{1.77}$$

where $v$ is the typical velocity of a star in the considered system. For simplicity, if we assume that the velocity distribution of the field stars is Maxwellian with dispersion $\sigma$, it is possible to obtain an explicit expression for $D\left[\left(\Delta v_{\parallel}^2\right)\right]$ and the equation (1.77) can be rewritten as

$$t_{rel} = \frac{v^2 \sigma X}{4\sqrt{2}\pi G^2 \widetilde{\rho}\widetilde{m} \ln \Lambda G(X)} \tag{1.78}$$

where $\widetilde{\rho}$ is the mean mass density of the field stars, $\widetilde{m}$ the mean stellar mass, $\ln \Lambda$ is the Coulomb logarithm, $X \equiv \frac{v}{\sqrt{2}\sigma}$ and $G(X)$ is a function that can be expressed as

$$G(X) = \frac{1}{2X^2}\left[\mathrm{erf}(X) - \frac{2X}{\sqrt{\pi}e^{-X^2}}\right]. \tag{1.79}$$

Assuming that the velocity of the test star is equal to the Maxwellian dispersion i.e. $v = \sqrt{3}\sigma$ and that $\frac{G(X)}{X}$ does not vary rapidly with $X$ it is possible to obtain the following expression which is less approximated than (1.75) because it does not assume, a priori, a stationary state or constant density for the field stars

$$t_{rel} \simeq 0.34 \frac{\sigma^3}{G^2 m \rho \log \Lambda} \ . \tag{1.80}$$

To obtain again the expression (1.75) from (1.78) we can proceed as follows. First of all let us assume that the considered system is in virial equilibrium. Using the formula (1.58) we get

$$\overline{v^2} = 2\alpha \frac{GM}{R} \ . \tag{1.81}$$

Substituting in equation (1.78) and approximating $\log \Lambda$ with $\log N$ we have

$$t_{rel} \simeq \left(\frac{2\alpha GM}{R}\right)^{\frac{3}{2}} \frac{1}{4\sqrt{6}\pi G^2 \rho m \log N} \ . \tag{1.82}$$

The total mass of the system can be written as $M = Nm$ and, assuming constant density, we can consider

$$\rho = \frac{Nm}{\frac{4}{3}\pi R^3} \tag{1.83}$$

therefore, substituting into the expression 1.82 we get

$$t_{rel} \simeq 2\frac{\alpha}{3}\sqrt{\frac{\alpha}{3}} \frac{X}{G(X)} \frac{N}{\log N} \frac{R^{\frac{3}{2}}}{GM} \tag{1.84}$$

which can be written in the form (1.75) using the expression (1.73) for the system crossing time

$$t_{rel} \propto \frac{N}{\log N} t_c \ . \tag{1.85}$$

## 1.3 The numerical solution of the $N$-body problem

### 1.3.1 The double divergence of the potential

As we discussed in the previous sections, the mathematical model of the classical gravitational $N$-Body problem remained unchanged since

Newton's epoch. Nevertheless, the numerical techniques apt to solve its mathematical scheme are required to be very sophisticated, fast and accurate, thing that, still nowadays, constitutes a challenge for astrophysicists, mathematicians and computer scientists. In fact, we are still very far to simulate, for example, a typical galaxy, containing $\sim 10^{11}$ stars, over a relaxation time, with an acceptable accuracy and in reasonable human times (even with the help of the most powerful supercomputers). The numerical solution of the $N$-body problem is a difficult task mainly because of the so called *double-divergence* of the two-body interaction potential. As we saw in section 1.2.1 the Newtonian potential between a point of mass $m_i$ and another of mass $m_j$ is given by

$$U_{ij} = \frac{Gm_i m_j}{|\mathbf{r}_j - \mathbf{r}_i|} \equiv \frac{Gm_i m_j}{r_{ij}} = U_{ji}, \tag{1.86}$$

where $\mathbf{r}_i$ and $\mathbf{r}_j$ are the position vectors of the $i$-th and the $j$-th star, $G$ is the gravitational constant and $r_{ij} \equiv |\mathbf{r}_j - \mathbf{r}_i|$ represents the Euclidean distance between the two particles. As *ultraviolet divergence* we mean the singularity in the $U_{ij}$ potential for very close encounters ($r_{ij} \to 0$); the *infra-red divergence* corresponds to a never vanishing pair-wise interaction. This double divergence leads to two immediate consequences:

1. close encounters ($r_{ij} \to 0$) yield to an unbound force between interacting stars ($F_{ij} \to \infty$) producing an unbound error in the relative acceleration;

2. the resulting force acting on each particle belonging to a generic $N$-body system requires summation over $N-1$ pair-wise contributions, yielding to an $O(N^2)$ computational complexity, which can be overwhelming whenever, as in the relevant astrophysical cases, $N$ is very large (for instance, $N \simeq 10^{11}$ for a typical galaxy).

Moreover, the evaluation of $r_{ij}$ is a computationally intensive operation; in fact, it requires the evaluation of the irrational function *square root*, based on iterative methods (one of them is the Newton-Raphson strategy), which need more than one floating point operation to be completed. There are several strategies to face the numerical difficulties posed by the functional form of the Newtonian potential. The UV divergence is often faced introducing a *softening parameter*, $\epsilon$, in the interaction potential which becomes

$$U_{ij} = \frac{Gm_i m_j}{\sqrt{r_{ij}^2 + \epsilon^2}}. \tag{1.87}$$

It corresponds to substitute point masses with Plummer spheres of scale length $\epsilon$ [82]. In this way, close encounters are smoothed but, of course, this is paid by a loss of resolution at spatial scales of order $\epsilon$ and below. Generally, it is used expressing the softening parameter as the average distance of a particle to its closest neighbour (that we will be denoted by $\langle d \rangle$) multiplied by a coefficient $\alpha \ll 1$. Given that we can write

$$\frac{4}{3}\pi R^3 \simeq \frac{4}{3}\pi N \langle d \rangle^3 \Rightarrow \langle d \rangle \simeq \frac{R}{N^{\frac{1}{3}}} \tag{1.88}$$

where $R$ is the system characteristic dimension, then, the expression

$$\epsilon = \alpha \langle d \rangle = \alpha \left( \frac{R}{N^{\frac{1}{3}}} \right) . \tag{1.89}$$

can be used to determine a reasonable value of $\epsilon$. The parameter $\alpha$ can be chosen arbitrarily (a value around $10^{-3}$ is reasonable) but, obviously, the smaller it is, the better the numerical solution at small scales is (even if a smaller time step is needed in order to not loose too much accuracy).

On the other hand, to reduce the $O(N^2)$ complexity it is possible to use approximation methods.

## 1.3.2 Numerical methods

### Evaluating the accelerations

During the last years, the algorithms and techniques to solve numerically the $N$-body problem and the hardware facilities have been significantly improved. A detailed description of the different numerical methods to solve the $N$-Body problem can be found in [39]. Here, in broad lines, we group the numerical $N$-body techniques in the following three categories, depending on the different ways to evaluate mutual forces:

1. *Direct summation* : the force acting on the particle $i$ is computed by the complete sum of the contributions due to all the other $N-1$ particles in the system, that is

$$\mathbf{F}_i = \sum_{\substack{j=1 \\ j \neq i}}^{N} G \frac{m_i m_j}{r_{ij}^3} (\mathbf{r}_j - \mathbf{r}_i) \tag{1.90}$$

   where $m_i$ and $m_j$ are the masses of the particles $i$ and $j$, $r_{ij}$ is the distance between particle $i$ and particle $j$ and $G$ is the gravitational constant. "Direct summation" represents the simplest method to implement but, at the same time, it can be considered the most accurate; nevertheless, its computational complexity is high (order of $N^2$), therefore it requires huge computing power to be successfully applied to big (large $N$) astrophysical systems. The best known codes based on this approach are `NBODY4`, mainly developed by Sverre Aarseth [3] , $\phi$-`GRAPE` [49], $\phi$-`GPU` [16], the $N$-body integrator included in the `STARLAB` environment [83], `MYRIAD` [58], `NBSymple` [29] and `HiGPUs` [28].

2. *Approximation schemes* : the direct sum of inter-particle forces is replaced by another mathematical expression lighter in terms

of computational complexity. To this category belongs, for instance, the so called *tree algorithm*, which was originally introduced by Barnes and Hut [13] and its computational complexity is of $O(N \log N)$. Greengard and Rokhlin [47] proposed in the field of molecular dynamics the so called *Fast Multipole Algorithm* (FMA), claiming for an $O(N)$ computing complexity, at least in quasi-homogeneous 2D particles distribution. Unfortunately, the deep comparison between the FMA and the tree code to evaluate gravitational forces performed by Capuzzo-Dolcetta and Miocchi [23], showed that FMA has, in 3D, the same $O(N \log N)$ computational complexity of the BH tree code and it is slower in both homogeneous and clumpy cases. An example of a modern tree code is `Bonsai` [14] but also `BRIDGE` [44] which simultaneously takes advantages from both the direct and the tree approach. Another example is `TreeATD` (tree-code with Adaptive Tree Decomposition) developed mainly by Miocchi and Capuzzo-Dolcetta [75]. Another kind of approximation scheme is that developed by Ahmad and Cohen [7]. Using this strategy, during "regular" steps a direct summation approach is used, but, more frequently, during "irregular" steps, only the force from neighbour particles is evaluated. The widely used codes `NBODY6` and `NBODY7` [77] use this scheme.

3. *Grid methods* : many codes are based on the solution of the Poisson's equation

$$\nabla^2 \phi(\mathbf{r}) = -4\pi G \rho(\mathbf{r}) \tag{1.91}$$

on a grid leading to a discretized force field (to solve the Poisson's equation, one of the quickest algorithms is the *Fast Fourier Transform* [53]). This kind of method (also known as *Particle Mesh method*) reduces, as the tree approach, the computational complexity at the expenses of the accuracy. As the tree algorithm, it is widely used in cosmological, large-scale simulations; one of the most known codes which implements the FFT in the solution of

the Poisson's equation, in a combination with a tree algorithm, is `GADGET2` [88].

We do not want to enter here in a discussion on advantages, disadvantages and peculiarities of the many different ways suggested to reduce the computational complexity of the $N$-body problem, we just state what is, almost unanimously, accepted: any of them induce some source of error, which can be systematic and not easily controlled. Anyway, it is worth noting that only the *direct summation* approach avoids approximation errors but, obviously, it demands high computing power. Throughout this work we will focus our attention on the direct summation approach because we developed our codes adopting this strategy.

### Advancing the solution over time

Besides the way to evaluate accelerations, a method to advance the solution over the time must be chosen and implemented. In principle, every method to numerically solve ordinary differential equations can be applied. Let us consider the $N$-body system formulated in (1.12); the most simple numerical method to advance the solution from $t = t_0$ to time $t = t_0 + \Delta t$ is the so called *Euler method* which is based on a first order Taylor expansion

$$\mathbf{r}\left(t_0 + \Delta t\right) = \mathbf{r}\left(t_0\right) + \mathbf{v}\left(t_0\right)\Delta t \tag{1.92}$$

$$\mathbf{v}\left(t_0 + \Delta t\right) = \mathbf{v}\left(t_0\right) + \mathbf{a}\left(t_0\right)\Delta t \tag{1.93}$$

where $\mathbf{a}$ represents the acceleration. The Euler method is globally a first order algorithm because its truncation error is proportional to $\Delta t^2$. It is very easy to implement but very inaccurate for the majority of the cases especially if a softening parameter is not included in the gravitational potential. Reducing the time step $\Delta t$ may improve the integration in terms of accuracy but the computing time increases too. The natural ex-

tension of this discussion would be to increase the order of the method including, for example, higher order terms in the Taylor expansion or to find another strategy to derive a different integration algorithm.

Unfortunately, the widely used *Runge-Kutta methods* do not constitute the appropriate solution to increase the accuracy of the $N$-body simulations. Using these kind of methods we would get higher precision of the integration but, at the same time, a very slow algorithm too. In fact, for example, using a 4th order Runge-Kutta algorithm, we have to evaluate $N^2$ mutual distances four times per time step which is not convenient in terms of ratio between accuracy and computing time. Moreover, the above discussed algorithms belong to the class of *explicit* methods that is, to evolve a system from the state A to a state B, they do not require information about other states except of A itself. In general, explicit methods, performing integrations over a quite long interval of time, tend to exhibit an unavoidable growth of the total energy. This error is due to both the cumulation of the numerical truncation error over multiple time steps and to the difficulty to treat close encounters where the system becomes, so called, *stiff*. A problem is said to be *stiff* when some of its involved amounts (like, in our case, positions, velocities and accelerations) change their per step values too fast, making the numerical solution correct only if very small integration steps are used.

### Symplectic integrators

Standard integrators are said to become *dissipative* and they exhibit incorrect long term behaviour not only because of the numerical problems that we have just pointed out but also because the "classical" schemes perform, step by step, non-canonical transformations (let us say from coordinates $(r_n, v_n)$ to $(r_{n+1}, v_{n+1})$) yielding to a slightly different Hamiltonian. To face this problem the *symplectic integrators* were introduced. To introduce this class of methods, first of all, we need to give some def-

initions. A *canonical transformation* is a change of coordinate which has the property of preserving Hamiltonian form of dynamics, that is, performing a coordinate transformation, for example, from $(u, v)$ to $(x, y)$, we will have

$$H(u, v) = K(x, y) = K(x(u, v), y(u, v)) \tag{1.94}$$

where $H$ and $K$ are, respectively, the old and the new form of the Hamiltonian function of the system. We have to underline that an *Hamiltonian system* is a dynamic system, having $n$ degrees of freedom, defined, by an *Hamiltonian function* ($H$) which satisfies the so called *canonical equations*

$$\begin{aligned} \dot{p}_k &= -\frac{\partial H}{\partial q_k} \\ \dot{q}_k &= \frac{\partial H}{\partial p_k} \end{aligned} \tag{1.95}$$

where $q_k$ is the $k$-th generalized coordinate and $p_k$ is called its momentum conjugate. This system, of $2n$ first-order differential equations, can be written in a form, called *symplectic*, which employs the matrix formalism, defining the column vectors $\mathbf{z}$ and $\partial H/\partial \mathbf{z}$, both of them having $2n$ components, and a $2n \times 2n$ square matrix $\mathbf{J}$ such that

$$z_i = q_i, \qquad z_{i+n} = p_i \tag{1.96}$$

$$\left(\frac{\partial H}{\partial \mathbf{z}}\right)_i = \frac{\partial H}{\partial q_i}, \qquad \left(\frac{\partial H}{\partial \mathbf{z}}\right)_{i+n} = \frac{\partial H}{\partial p_i} \tag{1.97}$$

$$\mathbf{J} = \begin{pmatrix} \mathbf{0} & \mathbf{1} \\ -\mathbf{1} & \mathbf{0} \end{pmatrix} \tag{1.98}$$

where $i = (1, 2, ..., n)$, $\mathbf{0}$ is the $n \times n$ matrix composed of vanishing elements and $\mathbf{1}$ is the $n \times n$ identity matrix. Using these new entries, we can write the system (1.95) in a more compacted form

$$\dot{\mathbf{z}} = \mathbf{J}\frac{\partial H}{\partial \mathbf{z}} \ . \tag{1.99}$$

It can be shown that

*the necessary and sufficient condition for a transformation $(q,p) \rightarrow (Q,P)$ to be a canonical one is that Jacobian matrix of the transformation ($\mathbf{\Lambda}$) is symplectic, that is*

$$\mathbf{\Lambda}^T \mathbf{J} \mathbf{\Lambda} = \mathbf{J} \tag{1.100}$$

*where it is known that*

$$\mathbf{\Lambda} \equiv \frac{\partial (Q,P)}{\partial (q,p))} \ . \tag{1.101}$$

Consider, for example, the already discussed Euler's explicit method. For simplicity we will refer to a one-degree-of-freedom system and so we may write

$$\begin{aligned} q_{n+1} &= q_n + p_n \mathrm{d}t \\ p_{n+1} &= p_n + f(q_n;t_n)\mathrm{d}t \end{aligned} \tag{1.102}$$

where $n$ indicates the $n$-th integration step and $f(q_n;t_n) = \dot{p}_n = -\partial H/\partial q_n$. From (1.102) we have

$$\Lambda = \begin{pmatrix} \frac{\partial q_{n+1}}{\partial q_n} & \frac{\partial q_{n+1}}{\partial p_n} \\ \frac{\partial p_{n+1}}{\partial q_n} & \frac{\partial p_{n+1}}{\partial p_n} \end{pmatrix} = \begin{pmatrix} 1 & \mathrm{d}t \\ \frac{\partial f}{\partial q_n}\mathrm{d}t & 1 \end{pmatrix} = \begin{pmatrix} 1 & \mathrm{d}t \\ \nabla_q f(q_n)\mathrm{d}t & 1 \end{pmatrix} \tag{1.103}$$

where it has been set $\partial/\partial q_n = \nabla_q$. Since this is a one dimensional problem, from (1.100) we argue that the necessary and sufficient condition for the matrix $\Lambda$ to be symplectic is that $\det(\Lambda) = 1$. In our case we have

$$\det(\Lambda) = 1 - \nabla_q f(q_n)\mathrm{d}t^2 = 1 \Rightarrow f(q_n) = \text{constant} \tag{1.104}$$

but, obviously, this does not happen, for example, for a generic $N$-body system, where $f(q_n; t_n)$ is the gravitational acceleration which, indeed, depends on particles space coordinates. Therefore, we have just proven that the Euler's explicit method is not symplectic. Symplectic integrators can be constructed thanks to Hamiltonian splitting. It can be verified that, if $H = H_1 + H_2 + ... + H_k$, then we may construct, at least, a first-order symplectic method by composition of $k$ coordinate changes. For example, if we consider a typical Hamiltonian function given by

$$H(\mathbf{q}, \mathbf{p}) = K(\mathbf{p}) + U(\mathbf{q}) \ , \tag{1.105}$$

where $K$ can be considered the Kinetic part of $H$ and $U$ is the potential part, the canonical equations (1.95) corresponding to $H_1 = K$ are

$$\begin{aligned} \dot{\mathbf{q}} &= \nabla_{\mathbf{p}} K(\mathbf{p}) \\ \dot{\mathbf{p}} &= \mathbf{0} \end{aligned} \tag{1.106}$$

while, the others are

$$\begin{aligned} \dot{\mathbf{q}} &= \mathbf{0} \\ \dot{\mathbf{p}} &= -\nabla_{\mathbf{q}} U(\mathbf{q}) \ . \end{aligned} \tag{1.107}$$

Integrating equations (1.107) and applying the resulting transformation to a generic point of phase space $(\mathbf{q}_n, \mathbf{p}_n)$ we obtain another point $(\widehat{\mathbf{q}}, \widehat{\mathbf{p}})$ such that

$$\begin{aligned} \widehat{\mathbf{q}} &= \mathbf{q}_n \\ \widehat{\mathbf{p}} &= \mathbf{p}_n - \nabla_{\mathbf{q}} U(\mathbf{q}_n) \Delta t \ . \end{aligned} \tag{1.108}$$

Now, integrating equations (1.106) and applying the transformation to the point $(\widehat{\mathbf{q}}, \widehat{\mathbf{p}})$ we obtain

$$\begin{aligned} \mathbf{q}_{n+1} &= \widehat{\mathbf{q}} + \nabla_{\mathbf{p}} K(\mathbf{p}_{n+1}) \Delta t \\ \mathbf{p}_{n+1} &= \widehat{\mathbf{p}} \ . \end{aligned} \tag{1.109}$$

Thus, eliminating $(\widehat{\mathbf{q}}, \widehat{\mathbf{p}})$ from (1.108) and (1.109) we may write

$$\begin{aligned} \mathbf{q}_{n+1} &= \mathbf{q}_n + \nabla_{\mathbf{p}} K(\mathbf{p}_{n+1})\Delta t \\ \mathbf{p}_{n+1} &= \mathbf{p}_n - \nabla_{\mathbf{q}} U(\mathbf{q}_n)\Delta t \ . \end{aligned} \tag{1.110}$$

So we have just obtained a first-order symplectic method which, often, is called *Euler's symplectic method*. The construction of high order symplectic methods is a harder task even if one can follow the just shown strategy. A detailed description and construction of higher order symplectic integrators can be found in Yoshida [98]. The code `NBSymple` [29] developed by some members of our group some years ago implements both a 2nd order symplectic integrator (symplectic Leapfrog) and a 6th order symplectic algorithm based on what obtained by Yoshida [98] in his work. The code `NBSymple` perform very well in terms of energy conservation especially if the 6th order integrator is used. For systems composed by $N \gtrsim 10^4$ stars since, for each integration step, the accelerations must be evaluated more than once (7 times for the 6th order algorithm), the evolution becomes very slow. Moreover, to reach a good degree of accuracy integrating close encounters between stars, the time step should be reduced significantly and the dynamical evolution is further slowed down.

### Hermite's schemes

The current state of the art of direct $N$-body simulations is represented by the class to which the *Hermite's integration schemes* belong. In particuar, the Hermite's 4th order scheme has been widely used in the past years to dynamically evolve $N$-body systems efficiently in terms of both computing time and accuracy of the final solution. To advance positions and velocities from time $t = t_0$ to $t = t_0 + \Delta t$, the classical Hermite's method uses Taylor expansions up to the third time derivative of the

acceleration. In other words this scheme is 4th order accurate which corresponds to the order of the Taylor expansion for velocities

$$\begin{aligned}\mathbf{r} &= \mathbf{r}_0 + \mathbf{v}_0\Delta t + \frac{1}{2}\mathbf{a}_0\Delta t^2 + \frac{1}{6}\dot{\mathbf{a}}_0\Delta t^3 + \frac{1}{24}\ddot{\mathbf{a}}_0\Delta t^4 + \frac{1}{120}\dddot{\mathbf{a}}_0\Delta t^5 + O\left(\Delta t^6\right)\\ \mathbf{v} &= \mathbf{v}_0 + \mathbf{a}_0\Delta t + \frac{1}{2}\dot{\mathbf{a}}_0\Delta t^2 + \frac{1}{6}\ddot{\mathbf{a}}_0\Delta t^3 + \frac{1}{24}\dddot{\mathbf{a}}_0\Delta t^4 + O\left(\Delta t^5\right)\ .\end{aligned}$$

In the classical scheme only the quantities $\mathbf{a}_0$ and $\dot{\mathbf{a}}_0$ are calculated using their exact mathematical expressions while the higher order derivatives $\ddot{\mathbf{a}}_0$ and $\dddot{\mathbf{a}}_0$ are approximated in order to reduce the computing effort. At the beginning of the time step it is possible to calculate $\mathbf{a}_{i,0}$ and $\dot{\mathbf{a}}_{i,0}$ through the well known formulas

$$\mathbf{a}_{i,0} = \sum_{\substack{j=1\\ j\neq i}}^{N}\mathbf{a}_{ij,0} = \sum_{\substack{j=1\\ j\neq i}}^{N} m_j\frac{\mathbf{r}_{ij,0}}{r_{ij,0}^3}, \tag{1.111}$$

$$\dot{\mathbf{a}}_{i,0} = \sum_{\substack{j=1\\ j\neq i}}^{N}\dot{\mathbf{a}}_{ij,0} = \sum_{\substack{j=1\\ j\neq i}}^{N}\left(m_j\frac{\mathbf{v}_{ij,0}}{r_{ij,0}^3} - 3\alpha_{ij,0}\mathbf{a}_{ij,0}\right)\ . \tag{1.112}$$

To obtain approximated formulas for the higher order time derivatives of the acceleration we can expand, by mean of Taylor polynomials, the acceleration and its first order time derivative

$$\mathbf{a_1} = \mathbf{a}_0 + \dot{\mathbf{a}}_0\Delta t + \frac{1}{2}\ddot{\mathbf{a}}_0\Delta t^2 + \frac{1}{6}\dddot{\mathbf{a}}_0\Delta t^3 \tag{1.113}$$

$$\dot{\mathbf{a_1}} = \dot{\mathbf{a}}_0 + \ddot{\mathbf{a}}_0\Delta t + \frac{1}{2}\dddot{\mathbf{a}}_0\Delta t^2\ . \tag{1.114}$$

From (1.114) we get

$$\ddot{\mathbf{a}}_0 = \frac{\dot{\mathbf{a_1}} - \dot{\mathbf{a}}_0 - \frac{1}{2}\dddot{\mathbf{a}}_0\Delta t^2}{t} \tag{1.115}$$

which substituted into the equation (1.113) let us write an expression for $\dddot{\mathbf{a}}_0$ as a function of only acceleration and its first derivative at the beginning of the time step and at its end

$$\dddot{\mathbf{a}}_0 = \frac{6\left[2\left(\mathbf{a}_0 - \mathbf{a_1}\right) + \left(\dot{\mathbf{a}}_0 + \dot{\mathbf{a_1}}\right)t\right]}{t^3}\ . \tag{1.116}$$

Similarly it is possible to obtain a similar expression for the second derivative of acceleration

$$\ddot{\mathbf{a}}_0 = \frac{2\left[-3\left(\mathbf{a}_0 - \mathbf{a_1}\right) - \left(2\dot{\mathbf{a}}_0 + \dot{\mathbf{a_1}}\right)t\right]}{t^2} \ . \tag{1.117}$$

Since equations (1.116) and (1.117) depend both on quantities which must be evaluated at the end of the time step, the Hermite's method, in its complete form, is implicit. Nevertheless, it is always possible to insert it in a *Predictor-Evaluation-Corrector* (PEC) scheme. At the beginning of the time step positions and velocities are predicted using the evaluated values of $\mathbf{a}_0$ and $\dot{\mathbf{a}}_0$ only, obtaining

$$\mathbf{r}_{i,pred} = \mathbf{r}_{i,0} + \mathbf{v}_{i,0}\Delta t + \frac{1}{2}\mathbf{a}_{i,0}\Delta t^2 + \frac{1}{6}\dot{\mathbf{a}}_{i,0}\Delta t^3 \tag{1.118}$$

$$\mathbf{v}_{i,pred} = \mathbf{v}_{i,0} + \mathbf{a}_{i,0}\Delta t + \frac{1}{2}\dot{\mathbf{a}}_{i,0}\Delta t^2 \ . \tag{1.119}$$

Using the values of $\mathbf{r}_{i,pred}$ and $\mathbf{v}_{i,pred}$ it is possible to compute $\mathbf{a}_1$ and $\dot{\mathbf{a}}_1$ (*evaluation step*) which can be used in combination with $\mathbf{a}_0$ and $\dot{\mathbf{a}}_0$ to obtain numerical values for equations (1.116) and (1.117). Then, the corrections

$$\mathbf{\Delta r}_i = \frac{1}{24}\ddot{\mathbf{a}}_0\Delta t^4 + \frac{1}{120}\dddot{\mathbf{a}}_0\Delta t^5 \tag{1.120}$$

$$\mathbf{\Delta v}_i = \frac{1}{6}\ddot{\mathbf{a}}_0\Delta t^3 + \frac{1}{24}\dddot{\mathbf{a}}_0\Delta t^4 \tag{1.121}$$

are added to the expressions for $\mathbf{r}_{i,pred}$ and $\mathbf{v}_{i,pred}$ (*Correction step*) to complete the integration step. Actually, it is possible to get also an approximated expression for $\ddot{\mathbf{a}}_1$ using a first order Taylor expansion

$$\ddot{\mathbf{a}}_1 = \ddot{\mathbf{a}}_0 + \dddot{\mathbf{a}}_0\Delta t \tag{1.122}$$

in order to improve the subsequent predictors. J. Makino was the first to formulate and to study this scheme in details (see [65] and [64]) generalizing some of the integration methods already developed by Sverre

Aarseth. Here we list some of the main advantages of the Hermite's scheme

1. the distances between particles have to be evaluated just once per integration step improving significantly the computing effort;

2. the correction step is very fast and stable;

3. despite the previous points the scheme is, globally, a 4th order algorithm;

4. the evaluation of accelerations and them first time derivatives can be done using dedicated machines (like `GRAPE`) or computing accelerators (like GPUs);

5. the scheme is easily adaptable to use individual (or block) time steps or other approximation schemes (like some neighbours strategy).

Regarding the latter point, one of the reasons for which the Hermite's scheme is, nowadays, widespread to numerically evolve $N$-body systems is that it can be efficiently used combined with a technique named *Block Time Steps* [69].

### Block time steps

It is trivial to understand why $N$-body systems are characterized by a wide range of time-scales corresponding to profound differences of trajectories, mutual distances and velocities of the stars inside the stellar system. In order to take into account such big interval it is convenient to transform the simple approach of *shared* to *individual* (per particle) time steps. In this way each star has his own time step which can be smaller or larger depending, in general, on the astrophysical parameters

of the star; for example, in the case of a close encounter with another member of the same system, the time step must be small enough to follow the numerical integration with an acceptable degree of accuracy. Specifically, using this approach, it is not needed to assign to all the stars of the system the same, very small, time step slowing down significantly the overall time evolution. It is enough to evolve, for each step, only the particles with smaller time steps reducing the computational complexity from $O(N^2)$ to $O(mN)$ if the number of stars to be updated in a step are $m$. The physical quantities of the other $N - m$ particles can be, in first approximation, left unchanged or predicted, using Taylor expansions, without evaluating their accelerations and higher order time derivatives till when they must be updated. Nevertheless, the use of individual time steps introduces some problems regarding the time synchronization of the $N$-body particles. In fact, requiring time synchronization between particles means that their time steps must be integer multiples each other but time steps are represented, in general, by real numbers (in single or double precision arithmetic). The main idea of the block time steps approach is to introduce hierarchical levels using the so called *quantization of time*: each particle is allowed to have a time step approximated using the closest power of two with (negative) integer exponent. This allows particles to be grouped into blocks which share the same time step, therefore stars belonging to the same level can be updated simultaneously also favouring the use of parallel accelerators. Each time step can be always reduced by a factor $2^\beta$ with $\beta > 0$ nevertheless, in order to maintain synchronization between particles and to guarantee better accuracy, a generic time step is not allowed to increase by a factor greater then $2$. Moreover, while the time step of a generic particle can be reduced, if needed, at every correction step, the increasing process cannot be always performed due to synchronization issues. To explain this latter point let us consider just 2 blocks, the first one having a time step $\Delta t_1$ and the other $\Delta t_2 = 2\Delta t_1$. Let us define also the *global time* $G_t$ of the simulation which must be advanced at each integration step and the *local time* $L_t$ which represents the time to which

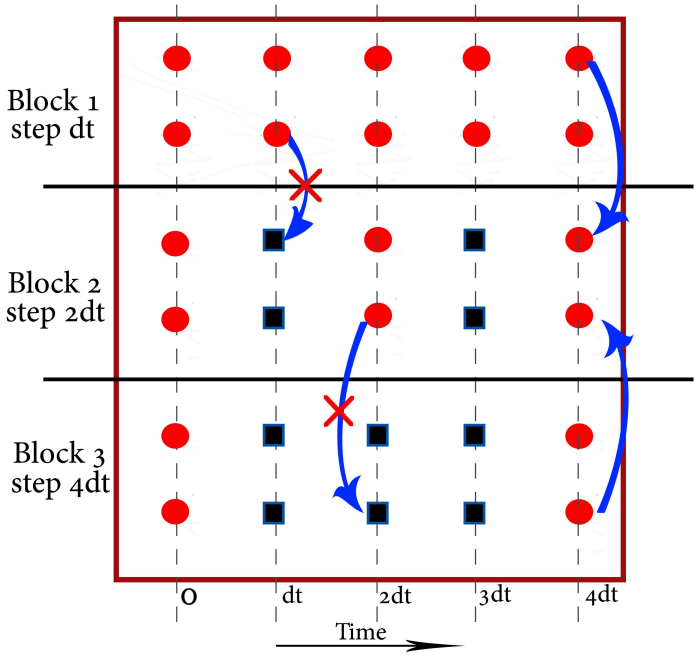


**Figure 1.4:** Schematic representation of the allowed (blue arrows) and forbidden (blue arrows with red corosses) transitions of particles (red spheres) in the block time steps strategy. Here we have considered 6 bodies distributed equally among three blocks; in particular, the 2 particles which belong to the block 1 have time step $dt$, those belonging to the block 2 have step $2dt$ and the last two stars in block 3 have time step equal to $4dt$. It is evident that, in this situation, the particles are synchronized at those times which are integer multiples of $4dt$. In these cases the stars are allowed to change freely their block. Transitions at intermediate stages, when some of the stars have only predicted physical position and velocity (black squares) must be performed carefully in order to avoid mismatches.

each particle has been evolved. Initially, $G_t = 0$, $L_{t,1} = 0$, $L_{t,2} = 0$. After we update the particles belonging to the first block we will have $G_t = \Delta t_1$ and $L_{t,1} = G_t$. At this time, the stars of the first block cannot be moved to the second block because particles are at two different

times: those of the first block are at time $L_{t,1} = G_t = \Delta t_1$ while the others are still at $L_{t,2} = 0$. The general condition such that particles can increase the time step by a factor $2^\beta$ and shift to the $\beta$ block is that

$$\left[\frac{G_t}{2^\beta \Delta t_{currentblock}}\right] - \frac{G_t}{2^\beta \Delta t_{currentblock}} = 0 \tag{1.123}$$

where the operator [] represents the integer part. This means that $G_t$ must be an integer multiple of $2^\beta$ times the current time step of the particle (in general $\beta$ is not allowed to be greater than 1). A schematic representation of the block time steps technique is shown in Fig, 1.4. For a typical $N$-body simulation about ten levels are populated and, in general, the time steps are allowed to change in a range between $2^{-2}$ and $2^{-25}$ (see, for example, Fig. 1.5). In order to determine the time steps of the stars different criteria have been experimented but this remains, by far, the most critical part to obtain, at the same time, an accurate and fast simulation. Simple criteria based on the requirement of slowly changing positions and/or velocities are proven to be not satisfactory for $N$-body simulations especially if used in combination of Hermite's methods and block time steps. Sverre Aarseth [4] stressed the importance to take into account also higher order time derivatives of the accelerations and, the requirement of small changes in the acceleration of the star $i$-th, yields to the criterion

$$\Delta t_i = \sqrt{\eta \frac{|\mathbf{a}|}{|\ddot{\mathbf{a}}|}} \tag{1.124}$$

where $\eta$ is a parameter introduced to control accuracy. The criterion expressed in equation (1.124) can be improved for the 4th order Hermite's algorithm considering also higher order time derivatives of the acceleration

$$\Delta t_i = \sqrt{\eta \frac{|\mathbf{a}|\,|\ddot{\mathbf{a}}| + |\dot{\mathbf{a}}|^2}{|\dot{\mathbf{a}}|\,|\dddot{\mathbf{a}}| + |\ddot{\mathbf{a}}|^2}} \tag{1.125}$$

which remains well defined also if a star starts from rest or if $|\mathbf{a}| \simeq 0$. Equation (1.125) expresses what is generally called the *Aarseth crite-*

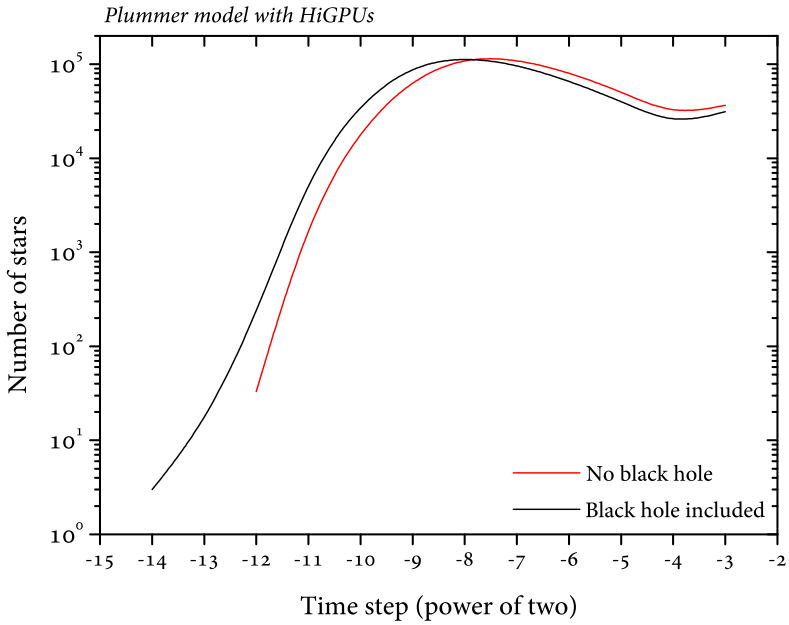


**Figure 1.5:** Time steps distribution obtained using our code `HiGPUs` integrating an isolated Plummer model with mass $M = 1$, scale radius $b = 1$ and $N \simeq 500,000$ (red curve). The black curve represents the same system but considering also the presence of a central super massive object with mass $M_{BH} = M$. It is evident that the inclusion of the central black hole expand the distribution curve toward smaller time steps increasing by a factor $\sim 2$ the speed of the numerical integration.

*rion* for $N$-body simulations.The Hermite's 4th order scheme has been also improved and generalized later by Keigo Nitadori and Junichiro Makino [78]. We report here the steps relative to the Hermite's 6th order scheme for a generic particle $i$:

1. Prediction step, with $O(N)$ complexity: positions, velocities and accelerations of all the stars are predicted using their known values:

$$
\begin{aligned}
\mathbf{r}_{i,pred} &= \mathbf{r}_{i,0} + \mathbf{v}_{i,0}\Delta t_{i,0} + \frac{1}{2}\mathbf{a}_{i,0}\Delta t_{i,0}^2 + \frac{1}{6}\dot{\mathbf{a}}_{i,0}\Delta t_{i,0}^3 + \\
&+ \frac{1}{24}\ddot{\mathbf{a}}_{i,0}\Delta t_{i,0}^4 + \frac{1}{120}\dddot{\mathbf{a}}_{i,0}\Delta t_{i,0}^5, \\
\mathbf{v}_{i,pred} &= \mathbf{v}_{i,0} + \mathbf{a}_{i,0}\Delta t_{i,0} + \frac{1}{2}\dot{\mathbf{a}}_{i,0}\Delta t_{i,0}^2 + \frac{1}{6}\ddot{\mathbf{a}}_{i,0}\Delta t_{i,0}^3 + \\
&+ \frac{1}{24}\dddot{\mathbf{a}}_{i,0}\Delta t_{i,0}^4, \\
\mathbf{a}_{i,pred} &= \mathbf{a}_{i,0} + \dot{\mathbf{a}}_{i,0}\Delta t_{i,0} + \frac{1}{2}\ddot{\mathbf{a}}_{i,0}\Delta t_{i,0}^2 + \frac{1}{6}\dddot{\mathbf{a}}_{i,0}\Delta t_{i,0}^3.
\end{aligned}
$$

2. Evaluation step, with $O(Nm)$ complexity (using block time steps): the accelerations of $m \leq N$ particles as well as their first and second time derivatives are evaluated using the above predicted data. The mutual interaction between the $i$-th particle and the remaining $N-1$ is described by the following relations:

$$
\begin{aligned}
\mathbf{a}_{i,1} &= \sum_{\substack{j=1 \\ j\neq i}}^{N} \mathbf{a}_{ij,1} = \sum_{\substack{j=1 \\ j\neq i}}^{N} m_j \frac{\mathbf{r}_{ij}}{r_{ij}^3}, \\
\dot{\mathbf{a}}_{i,1} &= \sum_{\substack{j=1 \\ j\neq i}}^{N} \dot{\mathbf{a}}_{ij,1} = \sum_{\substack{j=1 \\ j\neq i}}^{N} \left( m_j \frac{\mathbf{v}_{ij}}{r_{ij}^3} - 3\alpha_{ij}\mathbf{a}_{ij,1} \right), \\
\ddot{\mathbf{a}}_{i,1} &= \sum_{\substack{j=1 \\ j\neq i}}^{N} \ddot{\mathbf{a}}_{ij,1} = \sum_{\substack{j=1 \\ j\neq i}}^{N} \left( m_j \frac{\mathbf{a}_{ij}}{r_{ij}^3} - 6\alpha\dot{\mathbf{a}}_{ij,1} - 3\beta_{ij}\mathbf{a}_{ij,1} \right),
\end{aligned}
$$

where $\mathbf{r}_{ij} \equiv \mathbf{r}_{j,pred} - \mathbf{r}_{i,pred}$, $\mathbf{v}_{ij} \equiv \mathbf{v}_{j,pred} - \mathbf{v}_{i,pred}$, $\mathbf{a}_{ij} \equiv \mathbf{a}_{j,pred} - \mathbf{a}_{i,pred}$, $\alpha_{ij} r_{ij}^2 \equiv \mathbf{r}_{ij} \cdot \mathbf{v}_{ij}$, $\beta_{ij} r_{ij}^2 \equiv v_{ij}^2 + \mathbf{r}_{ij} \cdot \mathbf{a}_{ij} + \alpha_{ij}^2 r_{ij}^2$

3. Correction step with complexity $O(m)$: positions and velocities of the mentioned $m$ particles to be updated are corrected using the above evaluated accelerations and their time derivatives:

$$
\begin{aligned}
\mathbf{v}_{i,corr} &= \mathbf{v}_{i,0} + \frac{\Delta t_{i,0}}{2}\left(\mathbf{a}_{i,1} + \mathbf{a}_{i,0}\right) - \frac{\Delta t_{i,0}^2}{10}\left(\dot{\mathbf{a}}_{i,1} - \dot{\mathbf{a}}_{i,0}\right) + \\
&+ \frac{\Delta t_{i,0}^3}{120}\left(\ddot{\mathbf{a}}_{i,1} + \ddot{\mathbf{a}}_{i,0}\right), \\
\mathbf{r}_{i,corr} &= \mathbf{r}_{i,0} + \frac{\Delta t_{i,0}}{2}\left(\mathbf{v}_{i,corr} + \mathbf{v}_{i,0}\right) - \frac{\Delta t_{i,0}^2}{10}\left(\mathbf{a}_{i,1} - \mathbf{a}_{i,0}\right) + \\
&+ \frac{\Delta t_{i,0}^3}{120}\left(\dot{\mathbf{a}}_{i,1} + \dot{\mathbf{a}}_{i,0}\right).
\end{aligned}
$$

The individual time steps for $m$ particles are, thus, updated, by mean of the so called *generalized Aarseth criterion* [78]

$$
\Delta t_{i,1} = \eta \left( \frac{A^{(1)}}{A^{(p-2)}} \right)^{\frac{1}{p-3}}, \tag{1.126}
$$

where

$$
A^{(s)} \equiv \sqrt{\left|\mathbf{a}^{(s-1)}\right| \left|\mathbf{a}^{(s+1)}\right| + \left|\mathbf{a}^{(s)}\right|^2}. \tag{1.127}
$$

In Eqs. (1.126) and (1.127), $p$ represents the order of the integration method and $a^{(s)}$ is the $s$-th time derivative of the acceleration. In particular, if we use $p = 4$, we obtain again the standard Aarseth criterion of equation (1.125). A typical value of the parameter $\eta$ is around $0.6$ for the 6th order scheme.

This is the state of the art for direct summation $N$-body simulations considered for a theoretical point of view. The modern technologies and strategies to implement them on a (super) computer will be discussed in the next chapter.

# 2 The Graphics Processing Unit and CUDA

## 2.1 Historical introduction

As we saw in chapter 1 many algorithms and strategies have been developed in order to reduce the high computational complexity of the $N$-body problem. Specifically, the introduction of approximation schemes, in the past years, was compulsory because the computing power needed to run realistic direct summation $N$-body simulations was exceedingly large. All the runs, until $\sim$ 2006, were made using Central Processing Units (CPUs) or special-purpose machines but, nowadays, Graphic Processing Units (GPUs) are slowly replacing them to perform scientific simulations thanks mainly to the introduction of *Compute Unified Device Architecture* (CUDA). CUDA is, in essence, a collection of instructions that extend standard programming languages like C or Fortran in order to allow the user to program GPUs. It has been introduced by the nVIDIA corporation (between 2006 and 2007) and it is applicable only to nVIDIA GPUs. There is another language, younger than CUDA (its first public release went out in 2009) which can be used efficiently to write programs for a wide range of architectures, from smartphones to supercomputers, which is called *OpenCL*. In principle, a code written in OpenCL is portable in the sense that can run on a wide range of hardware (including GPUs of different brand) nevertheless it is less optimized for the single device and it is clear that, in general, performs slightly worse than CUDA when nVIDIA GPUs are used. Moreover, being younger and slightly more complicated to learn, is less widespread and it incorporates less functions and utilities. In any case, independently

from the programming language, which significantly helped, recently, to let the GPUs become easily handled by the common user, the idea itself of the so called *General Purpose computing on Graphics Processing Units* (GPGPU) is not really new in terms of time. Helped by the even growing popularity of graphically driven operating systems (Microsoft Windows), by the introduction (in 1992) of the OpenGL library to write 3D applications and by the release of the first videogames, the nVIDIA corporation released the GeForce 256 (October 11, 1999) which, for the first time, could perform some graphics calculations (transform and lighting) directly on-board. Before the advent of this kind of architecture, the listed calculations were left to the CPU therefore the GeForce 256 was also marketed as the first GPU. It was with the introduction in 2001 of the 3rd generation of the GeForce series that the information about each pixel on a screen became completely controlled by the programmer. In fact the GeForce 3 is remembered to be the first GPU with programmable pixel and vertex shaders. This intoduced also the idea that the arithmetic linked to the generic pixel (information about color, textures, antialiasing, etc etc ...) could be reinterpreted differently from a pure "graphics" point of view. Nevertheless, common users, among them scientists, had to learn graphics languages available at that time like DirectX or OpenGL to perform the discussed "trick" and this was the main limitation to the diffusion of GPUs thought as general computing accelerators. This is why CUDA and the first GPU supporting it (GeForce 8800 GTX) were introduced some years later, specifically between 2006 and 2007. The main innovation is that this GPU is the first model of unified shader architecture which means that, on the GPU, all computational units can handle any type of shading tasks which is the main idea of general-purpose computation. In fact, in the same period (about 2007), also the Tesla brand appeared; with the word *Tesla* the nVIDIA corporation indicated a special series of its hardware dedicated exclusively to perform general purpose computation on GPUs and they were initially based on the same chip of the GeForce GTX 8800 (G80) even if with more specific dedicated drivers.

## 2.2 The modern GPU architecture

Today, a modern GPU is organized with a certain number of highly threaded *Streaming Multiprocessors* (SMs) containing a certain number of *Streaming (or Scalar) Processors* (SPs) which often are also called *CUDA cores*. The G80 chip was composed by 16 SMs each with 8 SPs while the most recent Nvidia Kepler architecture (see for example Tesla K20X, GK110) has 15 SMX's (next generation SM) even if each SMX contains 192 SPs. The intermediate generation (Fermi) has 16 SMs each composed by 32 (up to 48) SPs.


**Figure 2.1:** The nVIDIA G80 architecture (top) and a detail its one Streaming Multiprocessor. Figure taken from [91]

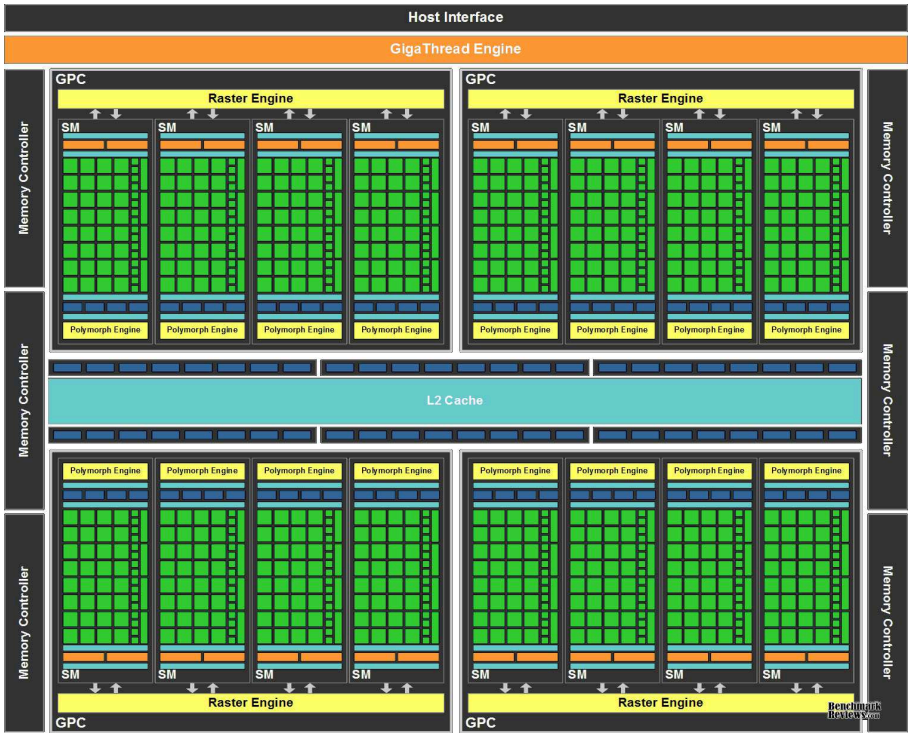


**Figure 2.2:** The nVIDIA Fermi architecture. Figure taken from [36]


**Figure 2.3:** The nVIDIA Kepler architecture. Figure taken from [37]


**Figure 2.4:** Differences between the Kepler (represented on the left) and the Fermi (on the right) Streaming Multiprocessors.

Figures 2.1, 2.2, 2.3 show the main internal hierarchical structure of three different nVIDIA architectures, which are respectively the G80 chip, the Fermi chip and the last generation scheme Kepler. In particular, Fig. 2.4 shows the main differences between the Kepler (represented on the left) and the Fermi (on the right) Streaming Multiprocessors. The CUDA programming model takes into account this hierarchical structure inside the GPU. In fact, in broad lines, the heart of any CUDA program is a function which is mapped on the GPU, called *kernel*. A kernel is a set of instructions which are executed simultaneously and independently by a certain number of virtual processing units called *threads*. The programmer must choice the number of threads to run and he organizes also all the threads in groups called *blocks*. The blocks are then organized in

other bigger groups called *grids*. The number of threads which can simultaneously run on a single GPU is a parameter which depends on the specific board and the number of threads per block and the dimension of the grid (1D, 2D or 3D) are strongly dependent on the nature of the algorithm to be implemented. Finding the ideal combination to maximize performance is a true challenge which requires a significant number of numerical experiments. In general, there is a correspondence between virtual units created by the programmer and physical components of the GPU; this reflects the internal structure seen in Figures 2.1, 2.2, 2.3, in fact each virtual thread is mapped to a single cuda core, each block of threads is mapped to a single SM (or SMX) and a grid is mapped on the entire GPU. Specifically, let us take, as example, a Tesla K20X. We will try here to obtain its main characteristics. The new Kepler GPU, as seen in Fig. 2.3, has 14 SMX's for a total of 2688 cuda cores. Hereafter, to measure and report the performance of a specific hardware we will use the value of *floating point operations per second* (`flops`). It is possible to calculate the maximum theoretical performance, in `flops`, that a Tesla K20X can perform. In fact, in each SMX, each cuda core can perform one single FMAD per clock. FMAD is the acronym of *Fused Multiply-ADd* which is equivalent to a single *Multiply-ADd* (MAD) floating point operation ($a + b \times c$) performed in one step with just one final rounding (corresponding to two floating point operations: one addition and one multiplication). Specifically, when a MAD calculates the product $b \times c$, rounds it to $k$ significant bits, adds the result to $a$, and rounds again to $k$ significant bits. A FMAD executes the entire operation rounding only the final result to $k$ significant bits improving performance calculating, for example, square roots (critical in solving numerically an $N$-body problem). With two floating point operations per clock cycle and a clock frequency of around 0.732 GHz per cuda core, having 2688 total cores we can evaluate the theoretical peak performance of the Tesla K20X as

$$0.732\,\mathrm{GHz} \times 2\,\mathrm{operations} \times 2688\,\mathrm{cores} = 3.94\,\mathrm{TFlops}\ . \qquad (2.1)$$

Nevertheless, this is valid for single precision operations which involve numbers stored in a memory space of 32bit (4 bytes). The Tesla K20X contains only 64 SPs, in each SMX, which can execute operations in double precision (64bit) therefore the theoretical maximum performance in 64 bit precision reduces to

$$0.732\,\mathrm{GHz} \times 2\,\mathrm{operations} \times 896\,\mathrm{cores} = 1.31\,\mathrm{TFlops}\ . \tag{2.2}$$

Moreover, Tesla K20X has six 64-bit memory partitions, for a 384-bit memory interface and it has 6 GB of GDDR5 DRAM memory. This important information is necessary to calculate, for example, the *theoretical maximum bandwidth* ($B_w$) which indicates the maximum amount of data that the GPU and its on-board memory can exchange in one second. Calling the memory interface $I_w$, having the memory clock frequency $\upsilon_M$, we can write

$$B_w(\mathrm{GB/s}) = \frac{2 I_w \nu_M}{8} = 0.25 I_w \nu_M \tag{2.3}$$

where $I_w$ is in bit, the factor $1/8$ converts $I_w$ in bytes, $\nu_M$ is in GHz and the factor $2$ takes into account that we have a DDR (*Double Data Rate*) RAM. For example, for a Tesla K20X we have

$$I_w = 384\,\mathrm{bit} \qquad \nu_M = 2.6\,\mathrm{GHz} \tag{2.4}$$

so, using the formula (2.3) , we obtain $B_w(K20X) = 250\,GB/s$.

Apart from the just described general properties of the so called global memory, which is also used to load and read data to and from the GPU, in the SMX we can find other kind of memories that have to be known in order to use them efficiently when we are approaching the implementation of a generic GPU code. Each SMX of the Tesla K20X contains 65536 *registers* of 32 bit. This is, by far, the fastest memory in the SM. Nevertheless, the user does not have direct control on register alloca-

tion; they are distributed among the threads by the compiler even if the programmer can induce their use using, for example, some built-in data types. The values stored in registers are local which means that registers are per-thread memory which is not visible to other threads on the GPU. There is also a certain amount of *local memory* but the access to this kind of memory space has high latencies and low bandwidth just like the global (device) memory space. According to the last version of the CUDA programming guide [34] the compiler stores automatically in local memory large structures or arrays, arrays for which it cannot determine that they are indexed with constant quantities but also single variables if the kernel uses more registers than those available. This latter behaviour, also known as *register spilling*, in general, should be avoided. There is also the *shared memory* space which has much higher bandwidth and much lower latency than local or global memory. There is a total of 64 kB of memory which can be configured and partitioned in shared memory and L1 cache. This kind of memory is shared within all the threads in a single block and is very important to optimize the performance of a generic $N$-body code. *Constant memory* is a space which resides in device memory and is cached in a constant read-only cache (48 KB in a Tesla K20X). It is used, in general, to broadcast a value of a read request to all threads that all refer to the same memory location.

In order to access memory (to read data or to store data), it is usually necessary to perform a simple calculation that returns a memory address. Each SMX of a Tesla K20X has 32 Load/Store units (LD/ST) which are dedicated to this purpose and they can calculate addresses for 32 threads per clock cycle. Moreover, 32 *Special Function Units* (SFUs) execute transcendental instructions, reciprocals, and square roots and are able to complete one instruction per thread and up to 4 floating point operations per clock cycle. The SMX schedules threads in groups of 32 virtual elements; these ensembles are called *warps*. Each SMX contains 4 *warp schedules* which, in broad lines, select and distribute instructions to cuda cores while each dispatch unit (8 in total) defines

their order of execution which is not necessarily ordered because warps are executed independently.

This concludes in broad lines all that we need to know, for the scopes of this work, about a modern GPU architecture in order to understand the implementation on GPUs of a generic $N$-body code. For a more detailed, technical and precise discussion about CUDA and GPUs we refer to [97], [86], [34], [36], [37], [35]. For more information about the OpenCL language we refer to [11], [48] and [76].

# 3 The $N$-body code HiGPUs

## 3.1 Motivation

The main purpose of this chapter is to introduce our new numerical implementation to solve the $N$-body problem using modern technologies like GPUs and the so called hybrid supercomputers that is very powerful machines composed by several computing nodes each containing one (or more) CPU connected through a PCI Express interface to one (or more) accelerator which can be a GPU. It is of primary importance to develop codes apt to work efficiently on hybrid machines because they are constantly growing in number, green efficiency (power consumption) and performance. In fact, looking at the Top500 list [95] which is the list of the most powerful supercomputers in the world, the second position belongs to `Titan` (see Fig. 3.1) [94] which harbours 18,688 nVIDIA Tesla K20X GPUs and at position number 10 we find the Tianhe-1A [93] which is composed by 7,168 nVIDIA Tesla M2050 GPUs but the list is constantly changing. Numerical codes which run natively on such big clusters are very rare because it is difficult to manage efficiently multicore CPUs, GPUs and more than one computing node all at the same time. Moreover, we decided to implement our own version of the numerical resolution of the $N$-body problem first of all because, in this way, we know perfectly how the code works and we know perfectly where to intervene if something goes wrong. This is a very important point when we need to deal with codes that can run on supercomputers especially if hardware accelerators like GPUs are present. In addition, there are no many $N$-body codes, freely available for the scientific community, which use GPUs and supercomputers, therefore when we need very powerful resources we do not have a wide choice. As we have already discussed,


**Figure 3.1:** This is a recent image of the Titan supercomputer, the second most powerful supercomputer in the world. It has a theoretical peak performance of ∼20 `Pflops` reached thanks to 18,688 nVIDIA Tesla K20 GPUs and ∼ 200,000 CPU (Opteron) cores and it occupies an area of more than 4,000 square feet.

the number of particles that can be integrated, over a reasonable interval of time (∼ 1 Gyr), using a modern direct summation $N$-body code is ∼ $10^6$. Another motivation to implement a new $N$-body code is to actively contribute to the modern research in order to increase this limit hoping that, as soon as possible, we will have numerical codes and technologies apt to model and dynamically evolve, for example, a typical galaxy (that is, $N \sim 10^{11}$). To our knowledge a $N$-body code similar to ours `HiGPUs` is $\phi$GPU which has been tested by their developers (see for example Berczik et al. [16]) even if its first official public release is not out yet, neither the related paper [17]. Some libraries (like `Sapporo` [45]), which are built to extend the compatibility of pure CPU $N$-body codes to GPUs, exist and are very useful and widespread but they do not have full support for the use of more than one computing node. Some other parallel GPU implementations of the famous $N$-body code `NBODY6` exist (see for example [77]) but do not support the use of hybrid supercomputers. Motivated by this we decided to implement out parallel $N$-body code `HiGPUs` guaranteeing natively full support for the modern hybrid architectures. The $N$-body problem is one of the real-world algorithms which is ideal to implement on GPUs. In fact, as we saw in chapter 2, GPUs are highly parallel machines which, nowadays, contain

thousands of cores able to execute hundreds of billion of floating point operations per second (`Gflops`) at the same time and independently from each other. Direct summation $N$-body codes, as we saw in section 1.3, are based on the all-pairs approach which has a very high computational complexity of $O(N^2)$ and the key idea is that the evaluation of accelerations can be done in parallel because the force acting on the $i$-th particle $\mathbf{F}_i$ is completely independent from that acting on the $j$-th particle $\mathbf{F}_j$ ($i \neq j$) because they depend only on positions which are always known at a certain time. In one of the first papers about the GPU implementation of the numerical $N$-body problem by Nyland et al. [79] we find in the conclusion section:

> *It is difficult to imagine a real-world algorithm that is better suited to execution on the G80 architecture than the all-pairs $N$-body algorithm.*

The paper by Nyland et al. [79] is very important because it introduced for the first time the efficient use of CUDA and GPUs to the numerical resolution of the $N$-body problem showing also, explicitly, techniques and pieces of CUDA code relative to their implementation on a now old nVIDIA GeForce GTX 8800 (G80) (already introduced in this work in chapter 2) obtaining very high performance ($\gtrsim 200$ `Gflops`) corresponding to a speed up of about a factor 100 with respect to other previous CPU-only implementations. Even if some modifications have been introduced in the main CUDA $N$-body kernels of modern codes, all of them reflect the strategies already introduced in [79].

## 3.2 Main features of HiGPUs

### 3.2.1 Parallelization scheme

Our direct summation $N$-body code is called HiGPUs [1] which stands for Hermite's integrator running on GPUs. The code implements a Hermite's 6th order integrator (see section 1.3), it is written combining tools of C and C++, it uses CUDA (or OpenCL, we have both versions) to exploit the power of GPUs and it is written using OpenMP [81] and MPI [80] to exploit the computing power of modern multicore CPUs distributed, in case, over many computational nodes. The Hermite's scheme is implemented using the already discussed technique of block time steps (see section 1.3) allowing both high precision and speed in the study of the dynamical evolution of very large ($N$ up to 8M) stellar systems. At this point it should be clear that the evaluation step is the most expensive section in terms of number of operations to execute and the prediction step comes after (see section 1.3). We found convenient to avoid communications between CPUs and GPUs through the PCI Express interface as much as possible therefore we decided to put all the sections of HiGPUs (Predictor, Evaluation and Corrector) on the GPU although the correction step is light in terms of operations to execute provided that the total number of particles to update ($m$) is significantly less than $N$ with $N$ not so high ($N \lesssim 500k$). The fundamental importance of the GPU in HiGPUs constitutes another novelty for $N$-body GPU codes. The parallelization scheme of the most recent version of HiGPUs is slightly different from that originally described in [28]. If $n_g$ is the number of GPUs used in a generic simulation of $N$ bodies, each GPU deals with the predictor of $\frac{N}{n_g}$ particles and evaluates $3m\frac{N}{n_g}$ accelerations and their first and second time derivatives. The accelerations

[1] Both the CUDA and OpenCL version of HiGPUs are freely available at http://astrowww.phys.uniroma1.it/dolcetta/HPCcodes/HiGPUs.html

are then collected and reduced by means of the `MPI_Allreduce()` functions. Then the corrector is performed on the GPU but not in parallel; in fact, each GPU executes the corrector for all the $m$ particles to update and determines their new time steps. This is done to avoid further possible bottlenecks due to communications between computing nodes and CPUs and GPUs inside the same node. Nevertheless, the immediate consequence is that, at the beginning of the next integration step, the same corrected particles could have slightly different values of positions and velocities depending on the way the single GPU has just executed the calculus. This is due mainly to round off errors which propagate, of course, to the following evaluations of accelerations and so on. To avoid further propagation, `HiGPUs` synchronizes again all the corrected values of positions and velocities every time all the $N$ particles must be update (that is when $m = N$). In this way the time needed to perform the latter broadcast operations becomes negligible if compared to that spent to execute the other sections of `HiGPUs` when $m = N$.

### 3.2.2 The `Bfactor` variable

Regarding the implementation of the evaluation step, some considerations about the maximum theoretical performance of the GPU have to be done. As we saw in broad lines in chapter 2 a Tesla K20X can to run up to 64 warps per SMX, which are 14, and each warp is a group of 32, virtual, GPU threads. Therefore, in principle, considering the resources available on the GPU, depending on the kernel that has to be executed, a Tesla K20X can run a maximum of 28,672 threads in parallel. To exploit the full power of this GPU is, therefore, necessary to run at least $28,672$ threads in parallel. If the stars to be updated are $m$ and the GPUs to use are $n_g$, the simplest, but not so efficient, parallelization scheme of the forces calculation would be such to run $m$ threads per GPU and calculate the partial accelerations (and their derivatives) due to $N/n_g$ bodies. This is straightforward but, if $m$ is small (less than $28,672$), there is not

enough work to fully occupy the GPU, causing a significant decay of the performance. In this low-$m$ regime, in order to increase the number of GPU thread blocks to map to as many SMX's as possible, `HiGPUs` tries to reduce the number of threads per block to use in the computation, starting from an a priori chosen value of 128[2] down, until this number reduces to the minimum possible, set to 32[3]. Anyway, this may be not enough to guarantee a good load of the GPU. To cope with this we introduced a variable, which we call *Bfactor*, acting as factor that multiplies, when necessary, the total number of GPU blocks of threads in order to split further the computation. For example, if $m < 28,672$ and *Bfactor* $= B$ we run $m * B$ threads per GPU and the partial accelerations (and derivatives) due to $N/(n_g B)$ stars are computed. Obviously, before passing the results to the CPU, each GPU has to deal with the *reduction* of $B$ blocks of accelerations. This adds a GPU reduction operation to our code but, as we will see, this cost is amortized by what is gained in the evaluation kernel. In any case, `HiGPUs` can recognize automatically the GPU in use, calculate the minimum number of parallel threads to fully load it, and determine the *Bfactor* maximum value by using the following procedure. Let us consider that, before the determination of the $B$ variable, we have a number of blocks to run ($L_0$) equals to

$$L_0 = \left[\frac{m}{\tau_L}\right] + 1 \tag{3.1}$$

where $\tau_L$ is the number of threads in a single block. The value of $B$ is determined by the formula

$$B = 2^{\left[\log_2 \frac{M}{\tau_L L_0}\right]+1} \tag{3.2}$$

[2]The value of 128 constitutes the result of many numerical experiments. We verified that, in almost all the practical situations, a number of threads per block greater than 128 does not produce a significant increase of performance.

[3]The value 32 represents a limit due to the size of the warp, that is the base unit to group virtual thread (as we already discussed in 2). For AMD GPUs this limit is increased to 64.

where $M = 28,672$ in the case of a Tesla K20X. The maximum values of the Bfactor variable ($B_{max}$) is fixed to

$$B_{max} = \frac{N}{n_g \tau_L} \tag{3.3}$$

where $n_g$ is the total number of GPUs that will be used in the simulation. It must be stressed that `HiGPUs` is built in order to work with values that are integer powers of two (number of particles, Bfactor value, number of computing nodes, time steps of the stars etc etc ...)[4].

### 3.2.3 Precision used in `HiGPUs`

Another main characteristic of `HiGPUs` is that it uses both single and double precision variables in the main GPU kernel (the evaluation of the forces). In fact, double precision is needed especially to calculate inter-particle distances where numerical cancellation errors might become critical parameters in determining the accuracy of the simulation. `HiGPUs` uses 64bit precision also to cumulate accelerations and their higher order time derivatives in order to reduce the error due to the propagation of the round-off errors on the single contributes (which are, at least, $N$ for each acceleration) which have to be cumulated on a single variable. All the other operations are performed in single precision, including the unavoidable square root which is calculated directly using the built-in function `rsqrtf()` that is an implementation of the reciprocal square root, which operates on single precision arguments, being significantly faster then the operation of $1.0/\mathrm{sqrt(a)}$ with an acceptable loss of precision. We adopted this approach because (as we saw in chapter 2) all the GPUs are significantly faster in performing single precision operations than executing 64 bit instructions (see chapter

[4] This strategy has been implemented for simplicity and convenience but a generalization is possible and it will be included in the next public release of `HiGPUs`

2) and, at the same time, a sufficiently high accuracy to evaluate forces is kept.

Anyway, also other approaches are found in the literature. For example, the use of emulated double precision or pseudo-double precision, (also called Double-Single, DS, precision) is widespread (see, for example [3] and [16]). In this way, only single precision operations are performed, replacing a 64-bit value with two, properly handled, 32-bit values. We have implemented a DS version of `HiGPUs` but we noticed that, although the performance was higher in terms of pure `Gflops` (as expected), the particle time steps distribution exhibited a sort of tail in the area of small time steps which is not present when using double precision to evaluate accelerations and higher order derivatives (this peculiar behaviour has already been pointed out by Gaburov et al. [45] comparing single and double-emulated precision). Therefore, using the DS version of `HiGPUs` , we obtain higher performance but a total execution time which is the same or greater and a relative energy conservation which is, on average, 2 orders of magnitude worse. A possible explanation of this behaviour is that `HiGPUs` uses the following criterion to determine particles time steps

$$\Delta t = \frac{1}{\alpha_1 + \alpha_2} \left( \alpha_1 \eta_4 \left( \frac{A^{(1)}}{A^{(2)}} \right) + \alpha_2 \eta_6 \left( \frac{A^{(1)}}{A^{(4)}} \right)^{\frac{1}{3}} \right) . \qquad (3.4)$$

This represents a weighted mean (with coefficients $\alpha_1$ and $\alpha_2$) between the Aarseth criterion for the 4th order Hermite's integrator (1.125) (with accuracy parameter $\eta_4$) and the generalized Aarseth criterion for the 6th order scheme (1.126) (with accuracy parameter $\eta_6$). The combination with $\alpha_1 = \alpha_2 = 0.5$ has been found to be more stable, for the 6th order method, than the two criteria used singly, providing better energy conservation and avoiding time steps too large or too small. The Aarseth style criteria are sensible to round-off errors and numerical terms cancellation in the calculation of higher order derivatives thus producing the above-said tail when using less precision to store variables. We do not discuss further this point here because it is out of the scopes of this

work, although a better investigation of this behaviour will likely lead to fix this problem in a future implementation of our code.

### 3.2.4 Tested architectures

An older version of `HiGPUs` has been deeply tested on a big hybrid supercomputer: the IBM iDataPlex DX360M3 Linux Infiniband Cluster (PLX) available, since June 2011, at the italian supercomputing consortium CINECA [32]. This is a supercomputer which consists of 274 computing nodes which exchange data through a Qlogic QDR (40 Gb/s) Infiniband high-performance network. Each node harbours 2 GPUs, for a total of 528 nVIDIA Tesla M2070 plus 20 nVIDIA Tesla M2070-Quadro, 2 CPUs Intel Xeon Esa-core Westmere E5645 running at 2.4 GHz and 46 GB of RAM memory. The operating system is Linux Red Hat EL 5.6 x86_64 while the version of the gcc compiler installed and tested is the 4.4.4, the CUDA version is the 4.0 and the OpenMPI version is the 1.4.3. Apart from the usage of large structures, we also tested the OpenCL version of our code on single different GPUs manufactured by different vendors (nVIDIA, AMD) obtaining surprising performance results. We will show that our OpenCL implementation of `HiGPUs` works fine on a wide range of GPUs which means that it is very portable and it is, to our knowledge, the first implementation in OpenCL of a $N$-body code, freely available, which can run on hybrid supercomputers. We report in this work these kinds of tests already published in two papers [28] and [25].

## 3.3 Results of performance tests on a hybrid supercomputer

In this section we show the results of our older version of `HiGPUs` both in terms of accuracy and performance performing some runs on the PLX

supercomputer. These test were of fundamental importance to get to the present version of `HiGPUs` because they let us understand where to intervene to significantly improve the performance of our code. For this purpose, we performed a set of $N$-body simulations with values of $N$ in the range from $32\text{k}$ to $8\text{M}$ stars, spatially distributed according to the Plummer mass density profile [82]

$$\rho(r) = \frac{3M}{4\pi b^3}\left(1+\frac{r^2}{b^2}\right)^{-\frac{5}{2}}, \tag{3.5}$$

where $r$ is the distance from the centre of gravity of the system and $b$ and $M$ are, respectively, the scale length (also called core radius) and the total mass of the system. The choices $b = 1$, $M = 1$ and, for the gravitational constant, $G = 1$ as units for the $N$-body simulations, lead to the system characteristic crossing time (cfr section 1.2.4) as unit of time for the code, written as

$$t_c = \frac{b^{\frac{3}{2}}}{\sqrt{GM}}. \tag{3.6}$$

At this regard, we note that there is no reason, a priori, to prefer other kind of units (like for example the so called $N$-body units, which we will briefly discuss in 4) to others even if, sometimes, they are simpler and more elegant. We used a softening parameter $\epsilon = 10^{-4}$, which is around 50 times smaller than the closest neighbour average distance (which scales as $N^{-1/3}$) for $N = 8\text{M}$. The choice of a fixed value of $\epsilon$ is not best suited to follow the, rare, very close encounters that may result in the formation of binaries but, for the scopes of our work, where we exploit our code's capabilities over relatively short time scales, it seems appropriate. Moreover, to perform our benchmarks, we chose values of $N$ as powers of two. This is not compulsory but apt to guarantee best performance of our code.

### 3.3.1 Energy and angular momentum conservation

The accuracy of our code, in its present version (not yet publicly released, still under tests) is controlled by the parameters $\eta_4$ and $\eta_6$ (see equation 3.4). Nevertheless, here we show the results obtained when the criterion of `HiGPUs` was just the generalized Aarseth criterion (i.e. $\eta_4 = 0$ and $\eta_6 = \eta$). To test the accuracy we run $N$-body simulations with $N = 2^k$ with $k$ integer between $[15; 20]$ over 10 time units, checking both the energy and the angular momentum conservation over that time interval. Specifically, the relative errors are calculated using the expressions

$$\Delta E_k = \frac{1}{10}\sum_{i=1}^{10}\left|\frac{E_k(t_i) - E_k(0)}{E_k(0)}\right| \qquad \Delta L_k = \frac{1}{10}\sum_{i=1}^{10}\left|\frac{L_k(t_i) - L_k(0)}{L_k(0)}\right|, \tag{3.7}$$

where $E_k(t_i)$ and $L_k(t_i)$ are, respectively, the total energy and absolute value of angular momentum of the $N = 2^k$ system evaluated at ten times $t_i$, which are multiples of the system crossing time. Moreover, the obtained values of $\Delta E_k$ and $\Delta L_k$ are averaged over the five values of $N$. Although this approach is quite arbitrary, we use it to estimate the accuracy of our code because it allows a comparison with, for example, the results obtained by mean of the direct $N$-body code by Berczik et al. [16] who used the same approach (although their values of $\eta$ are not comparable to ours because they use DS precision while our code uses separately single and double precision; for more details see section 3.2.3).

In Fig. 3.2 we show the results obtained for different values of $\eta$. As expected, the energy error does not depend on $\eta$ when $\eta$ is small enough; for $\eta \lesssim 0.3$ the relative energy error gets an almost constant value around $7.0 \cdot 10^{-11}$. Increasing the value of $\eta$ leads to a progressively

worse energy conservation because the particles time steps become, on average, larger, yielding to $\langle \Delta E_k \rangle \simeq 4.0 \cdot 10^{-3}$ for $\eta = 1.0$. A similar trend is noticed for the angular momentum error. We chose to maintain the energy error for our benchmarks below $5 \cdot 10^{-9}$ and the angular momentum error around $5 \cdot 10^{-7}$ so we set $\eta = 0.6$.

### Observations

We are actually testing the new criterion expressed in formula (3.4). The main problem is that the Aarseth criterion for the 6th order method allows, in general, particles to have bigger time steps than that obtained using the criterion for the 4th order method (already stressed by Nitadori and Makino [78]). Nevertheless, the 6th order criterion is also more affected by round-off errors and terms cancellation (the criterion for the 8th order even more than that for the 6th order) therefore, sometimes, especially when, for example, the acceleration of a generic particle is almost constant over a certain interval of time, (this can be verified for a star which escapes from the simulated stellar system), the time step tend to shrink significantly even if this reduction is not needed. The forth order criterion is less affected by this error, therefore it can be used together with other more elaborated criteria to control more accurately the distribution of time steps. We are currently conducing a deep study about this topic and the results will be shown in a forthcoming publication.

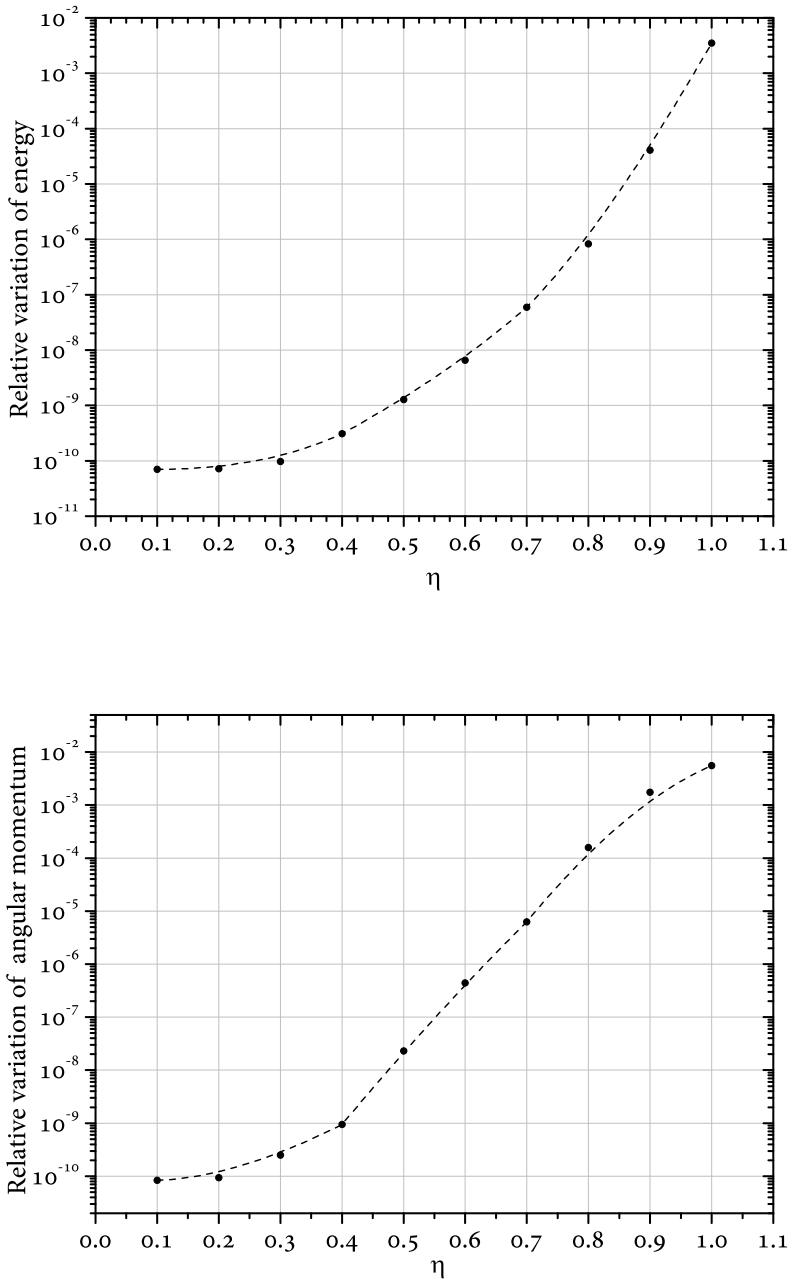


**Figure 3.2:** Averaged relative energy (upper panel) and angular momentum (lower panel) errors as a function of the accuracy parameter $\eta$.

## 3.3.2 Code scalability

Figure 3.3 shows the wall clock time needed to integrate an $N$-body system, for different values of $N$, up to one unit of time as a function of the number of GPUs used.

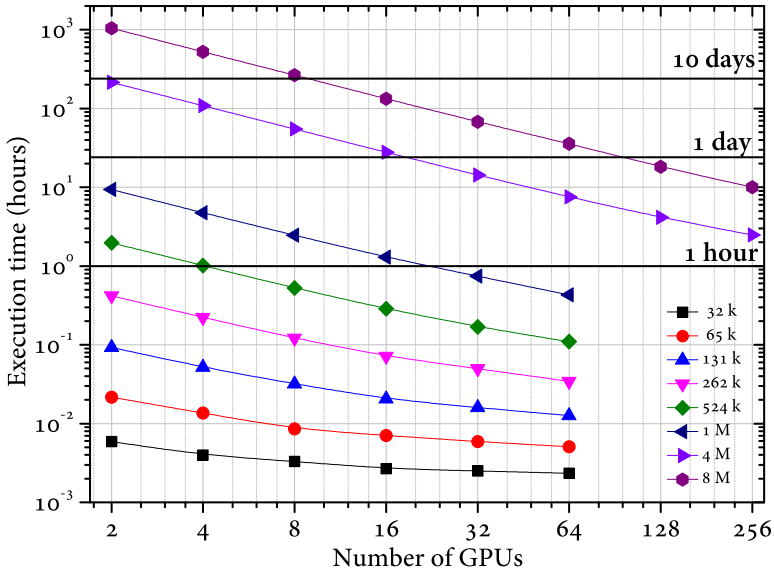


**Figure 3.3:** Time needed to integrate $N$-body systems ($32k \leq N \leq 8M$) over one time unit using different numbers of nVIDIA Tesla M2070 GPUs. Unfortunately, we could not use the PLX cluster to perform further simulations using 128 and 256 GPUs, therefore we do not have complete data for $N \lesssim$ 1M and a number of GPUs greater than 128. Nevertheless, this does not constitute a problem for the scopes of this work.

As relevant result, the total execution time decreases linearly increasing the number of GPUs whenever the number of bodies is large enough (1M, 4M or 8M). The departure from this inverse linear trend is seen

for $N \lesssim 262\text{k}$ and when the number GPUs used is greater than 16. This is expected because, when the number of particles per GPU is small ($\lesssim 1000$), the computational load is not enough to exploit the full computing power of the GPUs. Specifically, in this case, memory latencies, MPI communications, and other non-scalable parts of our code are not "covered" adequately. Another important output of Fig. 3.3 is that, using 256 GPUs, an integration of a 8M-body system over one time unit is done in less than 10 hours which is, to our knowledge, an unprecedented degree of performance for such kind of high precision, direct summation, $N$-body simulations.

#### Notes on the last version of `HiGPUs`

With the introduction of the new version of `HiGPUs` , thanks also to the introduction of always new functions and utilities in both CUDA and OpenCL, we improved further the results presented in Fig. 3.3 and now the departure from the linear trend is observed for $N \lesssim 65\text{k}$ when the number GPUs used is greater than 16 while the total times relative to the 8M and 4M bodies systems are reduced approximatively by a factor of about $1.4$ over all the x-axis.

### 3.3.3 Speedup and Efficiency

A deeper analysis of the performance of our code may be done by mean of the use of parameters like the *speedup* ($S_n$) and the *efficiency* ($E_n$). The speedup quantifies how faster a parallel algorithm is with respect to the corresponding sequential one, and it is defined as:

$$S_n = \frac{\Delta T_1}{\Delta T_n}, \tag{3.8}$$

where $\Delta T_n$ is the time spent to execute the program using $n$ computational units (GPUs, in our case). A parallel algorithm is considered to

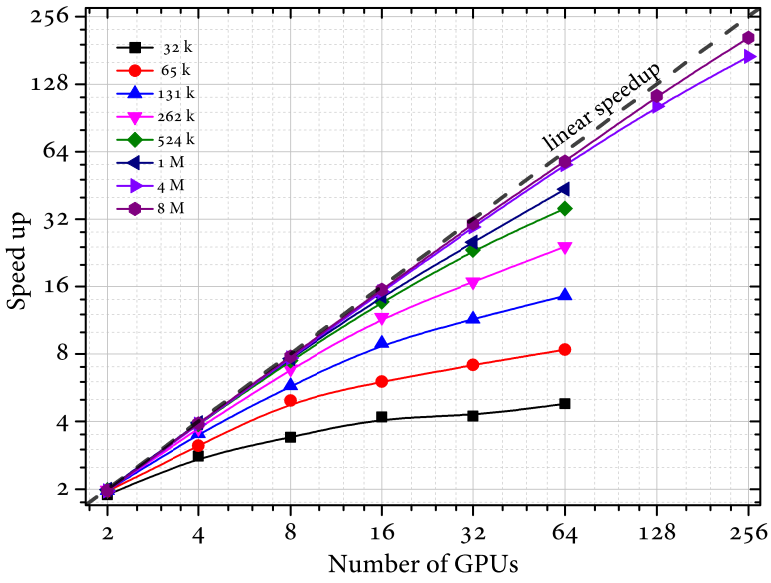


**Figure 3.4:** The speedup of our code as function of the number of GPUs used. The straight dashed line represents the trend of the perfect speedup.

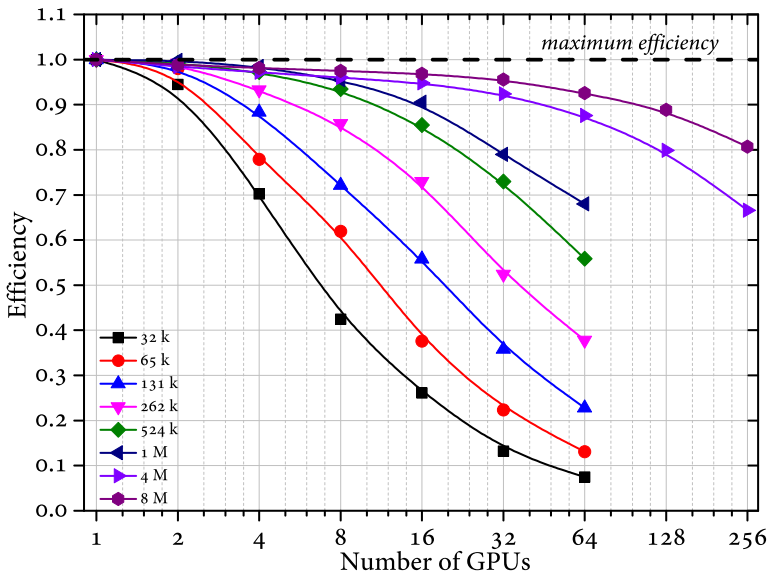


**Figure 3.5:** The efficiency of our code as function of the number of GPUs used. The horizontal dashed line represents the trend of the perfect efficiency.

be perfectly written if the so called *linear speedup* is reached and maintained increasing the number of computing units. This ideal situation corresponds to $S_n = n$.

A parameter which derives directly from the speedup is the efficiency of a generic algorithm, $E_n$, which indicates how the parallel algorithm exploits the whole available computing resources. It is usually expressed as:

$$E_n = \frac{S_n}{n}. \tag{3.9}$$

Low efficiencies mean a huge amount of time spent in data communications and/or synchronization events, that are, indeed, real bottlenecks for almost all highly parallel applications.

Figures 3.4 and 3.5 draw how our code is able to integrate up to 8M particles keeping a very good efficiency ($\simeq 0.80$) when using 128 nodes, decreasing to $\sim 0.70$ for $N \simeq 4$M. With the new modifications introduced in the new version of `HiGPUs` we get to a value $E_{256} \simeq 0.92$ for $N =$8M. Nevertheless, the smaller the number of bodies, the worse the scalability of our code is. This is a behaviour common to this kind of numerical codes, direct consequence of, at least, two different factors:

1. when the number of particles is small, as we said before, there is not enough work assigned to the generic GPU thread to cover adequately latencies. This can be due to different reasons

    a) to the too frequent GPU global memory access compared to the computational load;

    b) to the data transfer between GPUs and CPUs;

    c) to idle GPU cores yielding the performance of the generic GPU to very low levels.

2. increasing the number of GPUs and computational nodes implies the necessity to exchange and reduce data through a network connection, an operation which becomes important in terms of total execution time if the number of nodes in use is high (high latencies and low bandwidth whose speed is around 40 Gb/s for the IBM PLX cluster).

At the light of these observations, it is clear that the highest efficiency is reached whenever a right balance between the number of GPUs and the size (in terms of $N$) of the astrophysical problem is reached.

### 3.3.4 Code profiling

To obtain a clear picture of where the critical, non scalable, parts of our code are, we divided the operations and tasks performed in a single time step into 9 parts and we measured their execution times. The schematic representation of our code and its main tasks is given and explained in Table 3.1.

To investigate the performance of the individual sections of our code, we will focus on a system composed of about 1M stars ($N = 2^{20}$ to be precise) chosen as reference because it exhibits an average behaviour among all our benchmarks. Following the notation listed in Table 3.1, we report in Fig. 3.6 the fractional times spent to complete different sections of our code as a function of the number of computing nodes used. It is worth noting that the force calculation section of the code becomes less important in terms of execution time at increasing the number of nodes, reducing from about 100%, when using 1 node (2 GPUs) only, to 75% with 32 computing nodes (that is, 64 GPUs). Simultaneously, the relative contribution of the other code sections increases, especially the MPI communication part which goes from $\sim$0.2% with 1 node to $\sim$10% when we use 32 nodes. The same happens for the corrector step ($\sim$7.5% of the total time with 32 nodes). Figure 3.7 is also very helpful

| Index | Section | Used resource | Notation |
|---|---|---|---|
| 1 | Each node determines the stars to be updated and their indexes indexes are stored in an array named *next* containing *m* integer elements | CPU (OpenMP) | $\Delta t_{next}$ |
| 2 | Each node copies to its GPUs the array containing indexes of *m* particles and the predictor step of *N/n* stars is executed | GPU | $\Delta t_{pred}$ |
| 3 | Each node computes the forces (and their higher order derivatives) of *m* particles due to *N/n* bodies | GPU | $\Delta t_{eval}$ |
| 4 | Each node reduces the calculated forces and derivatives of *Bfactor* blocks | GPU | $\Delta t_{redu}$ |
| 5 | Each node adjusts conveniently the reduced values | GPU | $\Delta t_{repo}$ |
| 6 | The CPUs receive the accelerations from the GPUs | GPU $\rightarrow$ CPU | $\Delta t_{DtoH}$ |
| 7 | The `MPI_Allreduce()` functions collect and reduce accelerations from all the computational nodes | CPU(MPI) | $\Delta t_{mpi}$ |
| 8 | Corrector step and time step update for *m* stars | CPU | $\Delta t_{corr}$ |
| 9 | The reduced accelerations (and derivatives) and the corrected positions and velocities of *m* bodies are passed to the GPUs of each node | CPU $\rightarrow$ GPU | $\Delta t_{HtoD}$ |

**Table 3.1:** The main sections in which our code is divided. We indicate with $n$ the number of GPUs used in the computation. The "convenient adjustment" mentioned in the description of the 5th section of our code refers to the re-organization of the computed and reduced accelerations and derivatives in one array only (instead of three) to improve the performance of the subsequent data transfer from the GPU to the CPU. In this way we execute one bigger copy instead of three smaller ones.

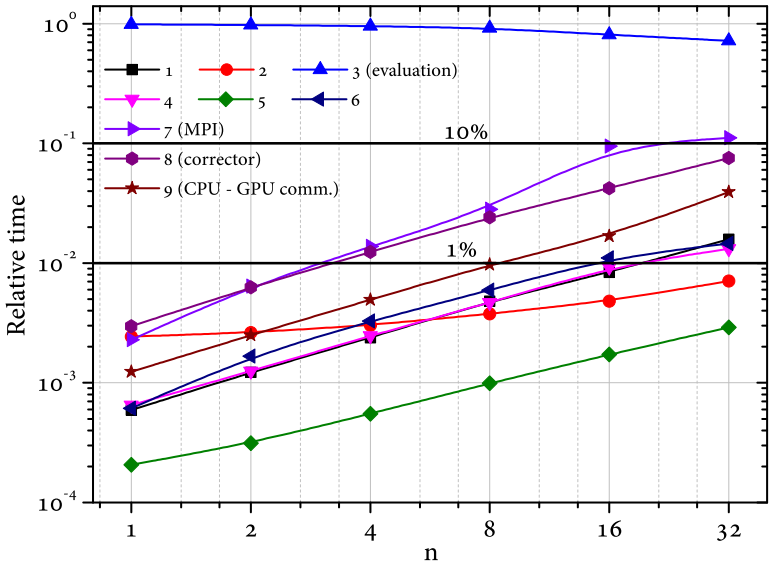


**Figure 3.6:** Relative (to the total) execution times of the different parts of our code (labeled as in Table 3.1) integrating an $N = 1M$ system up to one unit of time using different numbers of computing nodes.

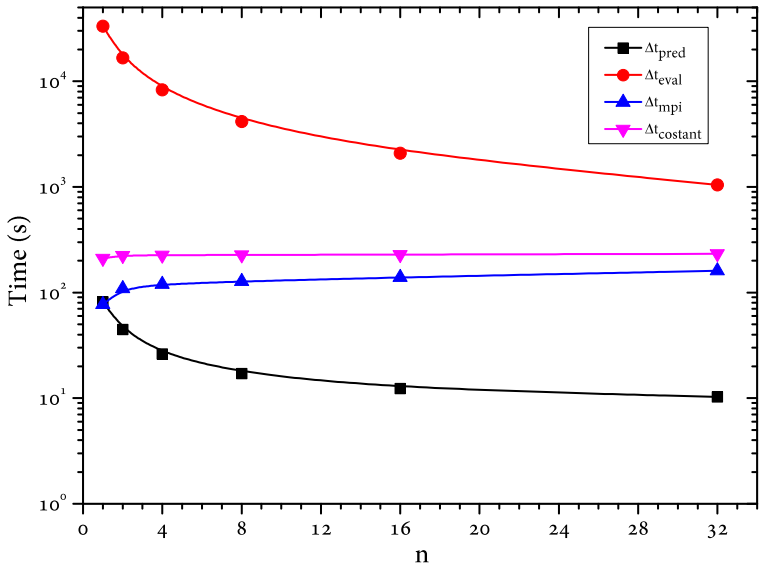


**Figure 3.7:** The times needed to complete the evaluation step, the predictor step, the MPI communications and the other sections of the code grouped together in the remaining curve. All the times refer to a system composed by 1M stars evolved over one time unit.

to identify possible bottlenecks in `HiGPUs` . It shows the time spent in the various sections of the code (as indicated in Table 3.1), as a function of the number of computational nodes, integrating the 1M-body system over 1 time unit. The figure shows that the two sections that scale with increasing the number of GPUs are the evaluation step (which is the most relevant part) and the predictor step.

The trend of the dependence of the evaluation step on the number of nodes, $n_n$, is well fitted by

$$\Delta t_{n_n,eval} = a \cdot {n_n}^{\alpha} \tag{3.10}$$

with best fitting values $a = 33,213.0 \pm 1.6\,\mathrm{s}$ and $\alpha = -0.99968 \pm 0.00030$. This denotes, according to Eq. 3.8, a very good speedup of our main gpu kernel, at least for $N$=1M and up to 64 GPUs. Moreover, we checked that for $N \in [32\mathrm{k}; 8\mathrm{M}]$ the value of $\alpha$ remains around $-1$, i.e. we obtain an approximatively linear speedup for the evaluation step. On the other hand, the $\Delta t_{mpi}$ part of the code, as shown in Fig. 3.7, grows with $n_n$ with an almost logarithmic scaling as a result of the combination of latency effects, low network bandwidth and inter-node reduction operations. The time spent in this part of the code is fitted by the expression

$$\Delta t_{n_n,MPI} = b + c \log_{10} n_n \tag{3.11}$$

with best parameters $b = 85.2 \pm 5.2\,\mathrm{s}$ and $c = 49.3 \pm 5.7\,\mathrm{s}$. The logarithmic growth of this section is common to all values of $N$ and may reduce significantly the efficiency and scalability of our code when using a large number of GPUs. At the light of the analysis above it is clear that a faster network connection could improve performance of our code significantly.

### Recent improvements

The figures shown in this section, already published in the paper by Capuzzo-Dolcetta et al. [28], helped us to understand the critical parts of our code. In fact, we noticed that, as we can see in Fig. 3.7, the corrector step constituted about 70% of the constant, not scalable part, of `HiGPUs` ; this is why, in the present version of `HiGPUs` , the corrector has been ported entirely on the GPU guaranteeing negligible times for this section with respect to other fundamental pats of our code. This gave us also the possibility to eliminate a further copy between the CPU and the GPU and following reorganization of positions, velocities and accelerations of the particles. In other words, the times relative to both sections 8 and 9 (see Tab. 3.1 and Fig. 3.6), are now negligible and

comparable to the times needed to perform the prediction step (code section 2).

### 3.3.5 Consequences of block time steps

Our direct summation $N$-body code is implemented by mean of hierarchically blocked time steps. This implies that the stars to be integrated in a generic time step may vary from 1 to $N$ depending on how the blocks are populated. As a consequence, it is interesting to see what are the groups of particles giving the biggest contribution in terms of the total execution time, in order to know where our code needs further optimization. To do this, we measured the update frequencies for the whole set of stars over one time unit (in our 1M-body reference case) using 32 computational nodes and then we multiplied these values by the sum of the times needed to complete each section of the whole time step for those bodies. After grouping particles into 6 groups (labeled with the letters A, B, C, D, E and F) we obtained the results sketched in Fig. 3.8.

Figure 3.8 show that, when using 32 nodes for the dynamical integration of 1M stars, the evaluation time can become significantly smaller, for example, than that for the per-block particles determination ($\sim$ 29% for the A group, brown part) or than that for the GPU reduction of the partial forces due to the presence of the *Bfactor* ($\sim$ 30% for the A group, in yellow). Moreover, the forces calculation time may be comparable to that needed to complete the predictor step (red) and to that needed to exchange data from the CPU to the GPU (pink). This situation becomes more evident when the number of nodes increases, while it fades, as expected, with larger number of bodies; the two effects tend to compensate each other. Therefore, in order to obtain a better performance, it is worth performing a further improvement of our code in the case when the number of stars to be updated is small compared to $N$. It must be

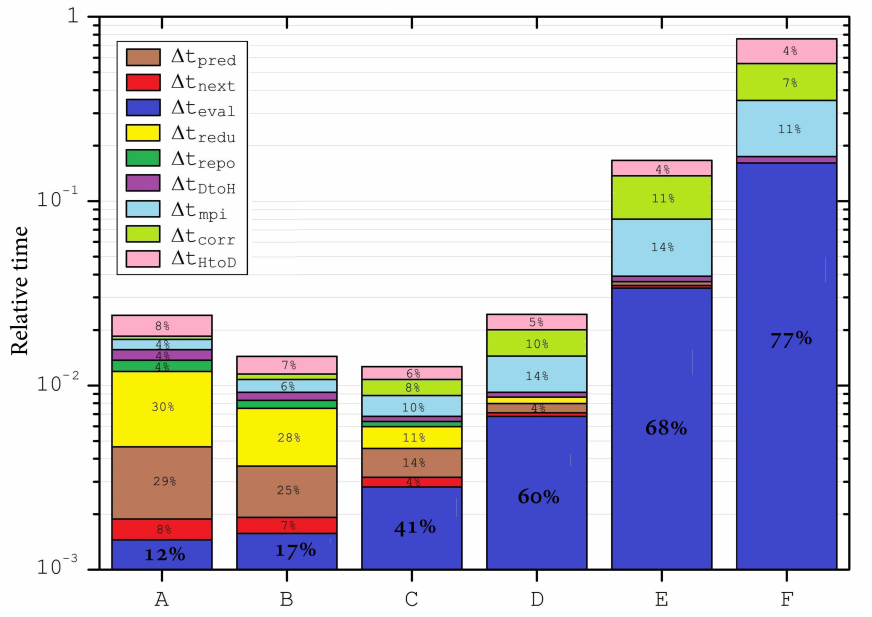


**Figure 3.8:** Considering the 1M-body reference system, calling $f_m$ the update frequency of $m$ particles over 1 time unit and calling $T_{m,s}$ the time needed to complete the section $s$ of our code for $m$ stars, the percentage value $H_G$ of each bar can be obtained as $H_G = \frac{100}{T_{tot}} \sum_s \sum_{m \in G} T_{m,s} \cdot f_m$, where $T_{tot}$ is the total time needed to complete 1 time unit and $G$ indexes the groups $A\,(m \in [1; 20])$, $B\,(m \in [21; 100])$, $C\,(m \in [101; 1k])$, $D\,(m \in [1k; 10k])$, $E\,(m \in [10k; 100k])$, $F\,(m \in [100k; 1M])$.

noted that these results are strongly related to the chosen initial conditions. In general, the overall performance of `HiGPUs` evolving a system containing $N$ bodies for a certain interval of time depends strictly on its speed for $m =< m >$ where

$$< m >= \frac{\sum_{i=1}^{S} m_i}{S} \tag{3.12}$$

where $S$ is the total number of integration steps and $m_i$ the number of particles to update for the $i-th$ step (Berczik et al. [16]). The value of $< m >$ depends on many factors like the distribution chosen to sample initial conditions, the value of the softening parameter, the criterion used for determining the particle time steps, the presence of a more massive body (black hole, see [26]) and also on the used time integration algorithm. A deep analysis about the relation which exists between $< m >$ and $N$ is now under investigation. In particular, in Fig. 3.8 we can see how, in our case, the majority of time ($\sim 80\%$) is spent avolving blocks of particles belonging to the group F, that is $m \in [100k; 1M]$. In any case, the use of the variable which we indicate as `Bfactor` (see section 3.2.2) improves significantly performance in critical (small $m$) regimes. In Fig. 3.9 we show the speed achieved in the forces calculation as a function of the number of bodies to be updated, having 16,384 stars per GPU, in the cases when the *Bfactor* optimization is active and when it is switched off. This mimics the situation in which an $N$-body system of 1,048,576 stars is integrated using 64 GPUs (32 PLX computational nodes). As we see in Fig. 3.9 the discussed optimization helps to improve performance up to a factor 50 when the number of particles to be updated is less then 20. This means that, referring to Fig. 3.8, the contribution to the total time of the first bar (A) would have been around 50 times larger without introducing a *Bfactor* value becoming the real bottleneck for our applications.

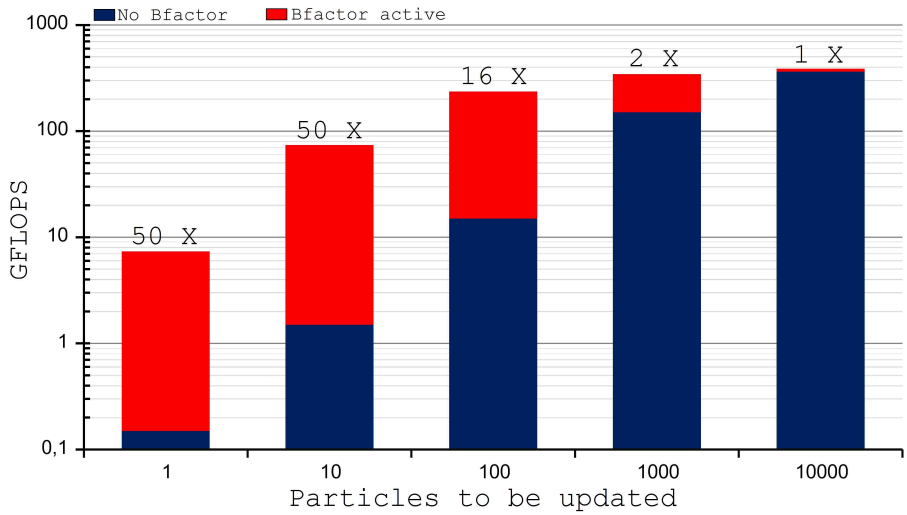


**Figure 3.9:** Speed (in `Gflops`) achieved updating different numbers of stars using 64 GPUs for a 1M-body system. The red bars indicate the improved performance when the *Bfactor* optimization is active.

### 3.3.6 GPU memory used by `HiGPUs`

As final benchmark, we investigated the maximum GPU memory used as a function of the number of stars to integrate. The results are shown in Fig. 3.10. As we can see, on a GPU Tesla M2070 it is possible to handle up to $N = 8$M stars, while, using a Tesla C2050, $N$ must be reduced to 4M.

This may seem, indeed, a real limit for our code applicability but it is important to stress that for real astrophysical direct $N$-body simulations (integrated over a significantly large interval of time, order of 1Gyr) it would be impossible to use, in practice, a number of stars $N \gtrsim 2$M because of the prohibitive execution times even using the most powerful supercomputing facilities available nowadays. Therefore the limited GPU on-board memory does not represent a real limit so far.

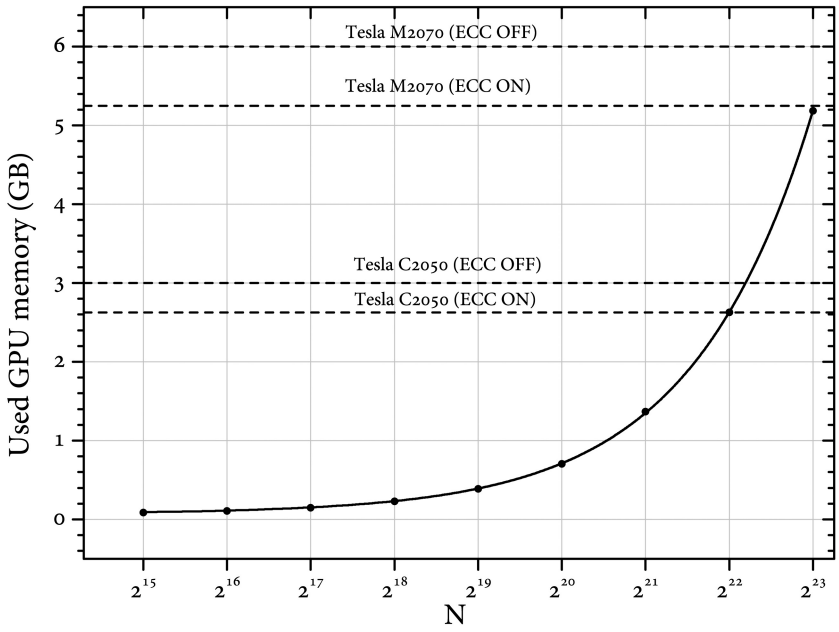


**Figure 3.10:** Used on-board GPU memory, in GB, as it grows with the number of stars of a generic $N$-body system.

### 3.3.7 Hardware maximum performance

As we have already discussed in chapter 2, the nVIDIA architecture named FERMI organizes the generic GPU as a group of 16 Streaming Multiprocessors of 32 Streaming Processors each for a total of 512 cores which often are called simply *cuda cores*. The GPUs tested in this work (Tesla M2070) are based on this kind of architecture having 448 active cuda cores over the 512 potentially available. Each of them has a clock frequency around 1.15 GHz and can execute up to two single precision floating point operations (32 bit) per clock cycle. This means that the theoretical performance peak in single precision can be determined by

$$S_{peak} = 448\,\text{cores} \cdot 2\,\text{flops/cycle} \cdot\ 1.15\,\text{GHz} = 1030.4\,\texttt{Gflops}. \quad (3.13)$$

| Operation | CUDA expression | Equivalent flops |
|---|---|---|
| $a \pm b$ | a ± b | 1 |
| $a \cdot b$ | a ∗ b | 1 |
| $\frac{1}{\sqrt{a}}$ | rsqrt(a) | 4 |
| $\frac{a}{b}$ | a/b | 5 |
| $a^b$ | pow(a,b) | 9 |

**Table 3.2:** The number of floating point operations required by the operations most relevant for our code.

On the other hand, up to 32 double precision floating point operations (64 bit) can be performed by each Streaming Multiprocessor, per clock cycle. Having 14 active multiprocessors on a Tesla M2070 we get

$$D_{peak} = 14\,\text{multiprocessors} \cdot 32\,\text{flops/cycle} \cdot\ 1.15\,\text{GHz} = 515.2\,\texttt{Gflops}, \tag{3.14}$$

which is exactly half of the single precision peak. To estimate how much our code can exploit the FERMI architecture we measured its peak performance. To do this, we counted how many floating point operations are enclosed in our evaluation kernel (the most expensive section in terms of computational load) and then we divided this value by the time needed to execute it, obtaining the performance expressed in `Gflops`. Other authors use different strategies to count operations [41, 79] but we prefer to refer to Table 3.2. Table 3.2 has been built following the Table 2 (Throughput of Native Arithmetic Instructions) shown at paragraph 5.4.1 (Arithmetic Instructions) in the document CUDA C programming guide [34] coupled with the information given by the whitepaper of the FERMI architecture [36]. Specifically, it is possible to stress that the power elevation operation is very expensive, and it must be avoided, as much as possible, because it is implemented using a combination of one base-2 logarithm, one base-2 power elevation and one multiplication. Following Table 3.2 we counted 15 double precision operations

plus 82 single precision operations in our main kernel. Therefore, we estimate the theoretical peak achievable, per GPU, by the formula

$$R_{peak} = \frac{82 \cdot S_{peak} + 15 \cdot D_{peak}}{82 + 15} \simeq 950\,\texttt{Gflops}. \tag{3.15}$$

This is, obviously, an ideal value because it does not consider any kind of memory latency, communication and/or read and write operations which, in general, can reduce performance significantly. The formula that we derived to count `Gflops` in our main kernel is the following

$$R \simeq \frac{97 \cdot N \cdot m}{10^9 \cdot \Delta T_{ker}(N, m)}(\texttt{Gflops}) \tag{3.16}$$

where $N$ is the total number of stars that form our $N$-body system, $m$ is the number of particles to be updated and $\Delta T_{ker}(N, m)$ is the kernel execution time. A similar formula has been used by other authors [78, 89]. We reached performance over 100 `Tflops` using 256 Tesla M2070 with $N = 2^{23} \simeq 8.4 \times 10^6$ stars, which corresponds to $\sim 400$ `Gflops` per GPU, that is around 40% of the claimed peak GPU performance. This is a very good result for this kind of astrophysical computations, especially in this case of heavy use of the double precision arithmetic, and is, at least, comparable to what obtained by other authors with similar (in structure) $N$-body. codes (see [16, 89]). As a final note, we stress that, in addition to the availability of powerful software and hardware tools, to reach higher performance in real physical/astrophysical applications that require simulations extended over long time scales, the development and improvement of the numerical algorithms is, in any case, unavoidable.

## 3.4 Final observations

It is now clear that composite architectures based on several computing nodes hosting multicore Cnetral Processing Units connected to one or

more Graphic Processing Units represent a clever and efficient solution to supercomputing needs in different scientific frameworks. Actually, these architectures are characterized by a high ratio between performance and both installation cost and power consumption. A practical proof of this is that some of the most powerful systems in the Top500 list of world's supercomputers [95] are based on that scheme. They are, indeed, a valid alternative to massively parallel multicore systems, where the final computational power comes by the use of a very large number of CPUs, although each of them has a relatively low clock frequency. It is quite obvious that a full exploit of the best performance of the CPU+GPU platforms requires codes that clearly enucleate a heavy computational kernel, to be assigned in parallel to the GPUs acting as "number crunchers" which release, periodically, their results to the hosts. In physics, the study of the evolution of systems of objects interacting via a potential, depending on their mutual distance, falls into this category.

In this chapter we presented and discussed a new, high precision, code apt to simulating the time evolution of systems of $N$ point masses interacting with the classical, pair, Newtonian force. The high precision comes from both the evaluation by direct summation of the pairwise force among the system bodies and by a proper treatment of the multiple space and time scales of the system, which means resorting to an individudal time-stepping procedure and resynchronizations, as well as using a high order (6th) time integrator.

We also discussed the implementation of our fully parallel version of a direct summation algorithm whose O($N^2$) computational complexity is dealt with by GPUs acting as computing accelerators in the hosting nodes where multicore CPUs are governed and linked via MPI directives. The code, called `HiGPUs` , available to the scientific community at the web address in footnote[5] or in the frame of the AMUSE project

[5] `astrowww.phys.uniroma1.it/dolcetta/HPCcodes/HiGPUs.html`

on amusecode.org [68], shows a very good performance both in term of scaling and efficiency in a good compromise between precision (as measured by energy and angular momentum conservations) and speed. Moreover, as discussed, the last version of HiGPUs (still under tests) has been further improved. We performed an extensive set of test simulation as benchmarks of our code using the PLX composite cluster of the CINECA Italian supercomputing inter-university consortium. We found that the integration of $N = 8,000,000$ bodies is done with an 80% efficiency, that is a deviation of just 20% from the linear speedup when using 256 nVIDIA Tesla M2070. This corresponds to less than 10 hours of wall clock time to follow the evolution of the 8M body system up to one internal crossing time, performance, at our knowledge, never reached for such kind of simulations.

This means that with HiGPUs it is possible to follow the evolution of a realistic model of Globular Cluster (a spherical stellar system orbiting our and other galaxies and composed by about 1,000,000 stars packed in a sphere of about 10 pc of radius) with a 1:1 correspondence between number of real stars in the system and simulating particles. Actually, as we will see in the next section, by mean of 4 AMD Radeon HD7970 GPUs, the GC dynamical evolution over 1 orbital revolution around the galactic center (at a galactocentric distance of about 8 kpc, i.e. the solar galactocentric distance) takes $\simeq 28$ days. This time should be scaled by a factor 3 when using the same number (4) of nVidia Tesla M 2070 GPUs.

These kinds of simulations will allow, for instance, a thorough investigation of open astrophysical questions that may involve, in their answer, the role of globular clusters and globular cluster systems in galaxies. We cite the open problem of the origin of Nuclear Clusters as observed in various galaxies, like our Milky Way. Some authors (e.g., [74] and [15]) suggested a dissipational, gaseous origin while others ([22], [27]) indicate, more realistically, a dissipationless origin by orbital decay and

merging of globular clusters, (hypothesis already numerically tested in [24] and [12]).

One limit in the use of our code is the GPU memory: with a 6GB RAM, as in the case of nVIDIA Tesla M2070, the upper limit in $N$ is $\sim 8,400,000$, which is, anyway, a number sufficiently large to guarantee excellent resolution in the simulation of most of the astrophysically interesting cases involving stellar systems.

Apart from these technical, hardware, considerations, we wish to remark that simulations extended over time length where secular behaviours deploy likely rely on a clever strategy of balance between computational power and algorithmic development which help, in any case, to have physically reliable results without necessarily resort on brute force computations.

We acknowledge the class C grant number HP10COVQZA provided by the Italian consortium for supercomputing (CINECA, Casalecchio, Italy) which allowed us to perform the simulations presented in this section.

## 3.5 HiGPUs on single, different GPUs

As we have already discussed in the previous chapters, nowadays, although the use of GPUs to accelerate $N$-body codes is widespread, very few codes have been implemented using OpenCL being very young and less optimized than CUDA. Therefore, the theoretical computing power of, for example, AMD GPUs has not been fully tested and compared with the performance of nVIDIA GPUs. A deep knowledge of hardware facilities and how to use them efficiently, as we discussed in chapter 1, is a fundamental point for modern scientists and researchers who want to use recent technologies to produce important scientific (numerical) results. The following section of this work (which has been already pub-

lished, see Capuzzo-Dolcetta and Spera [25]) focuses on the description and benchmarks of a wide range of GPUs in order to give an idea to how and what has to be used to perform a specific (in our case astrophysical) simulation. Although we will use our code `HiGPUs` , already introduced and deeply discussed in the previous section, the benchmarks presented in this section can be easily seen in a more general way as tests of a generic algorithm which has a computational complexity of $O(N^2)$, which represents an amount of floating point operations widespread in modern GPGPU applications.

Specifically, in this chapter, we will test the performance of the OpenCL version of `HiGPUs` to exploit and compare the measured computing power of the modern GPUs available on the market. We will show and discuss the comparison among different GPUs running `HiGPUs` to evolve several $N$-body systems corresponding to different astrophysical situations, chosen as test cases. Although we have a CUDA version of our code (whose tests have been shown in the previous chapter), in this section we need to use the OpenCL version to compare AMD and nVIDIA GPUs using an identical high-level software. It is worth underlining that the gain of performance running the same test cases on nVIDIA GPUs using the CUDA version of `HiGPUs` , instead of the OpenCL, has been quantified about 5 %. We introduced more recent improvements of the CUDA version of `HiGPUs` obtaining a significant gain with respect to the version used to realize the following tests. In fact, preliminary tests show that the gain of $\sim$5% is increased to $\sim$20%.

Firstly we will focus our attention to the different hardware (GPUs) tested; then we will describe how we measured performance and the astrophysical test cases. Finally we will show the results of the performed tests.

| GPU Model | Cores (number) | Clock (GHz) | 32bitP (Gflops) | 64bitP (Gflops) |
|---|---|---|---|---|
| AMD Radeon HD 6970 | 1536 | 0.880 | 2703 | 675 |
| AMD Radeon HD 7970 | 2048 | 0.925 | 3789 | 947 |
| AMD Radeon HD 7870 | 1280 | 1.000 | 2560 | 160 |
| nVIDIA GeForce GTX 580 | 512 | 1.544 | 1581 | 198 |
| nVIDIA GeForce GTX 680 | 1536 | 1.006 | 3090 | 129 |
| nVIDIA Tesla K20 | 2496 | 0.706 | 3520 | 1170 |
| nVIDIA Tesla C2050 | 448 | 1.150 | 1030 | 515 |
| nVIDIA Tesla C1060 | 240 | 1.300 | 622 | 78 |

**Table 3.3:** Some characteristics of each of the tested GPUs. The columns 32bitP and 64bitP list the maximum theoretical performance in single and double precision respectively.

### 3.5.1 Hardware

Our performance tests were done on different GPUs manufactured by nVIDIA and AMD corporations. In Tab. 3.3 and Tab. 3.4 we list the GPUs used for our benchmarks with some useful reference data.

It is possible to see in Tab. 3.3 that we used 3 GPUs of the AMD Radeon series, 2 GPUs of the nVIDIA GTX series and 3 GPUs of the nVIDIA Tesla series. It is important to stress that while GeForce and Radeon cards are explicitly designed for the gaming market, the Tesla cards are dedicated to scientific users and, since double precision operations are not needed for playing video games, both GeForce and Radeon cards have limited 64-bit computing capability. At this regard, we notice from Tab. 3.3 that the GTX 580 GPU has a double precision peak limited to 0.125 times its single precision speed, while this ratio is up to 0.5 for the Tesla C2050. Unfortunately, for the more recent card, the GTX 680, which is based on the so called Kepler architecture (see chapter 2), this factor is even smaller (about 0.1).

| GPU Model | Launch (quarter year) | TDP (Watt) | Memory (MB) | Bandwidth [a] (GB/s) |
|---|---|---|---|---|
| AMD Radeon HD 6970 | Q4 2010 | 250 | 2048 | 176.0 |
| AMD Radeon HD 7970 | Q1 2012 | 250 | 3072 | 264.0 |
| AMD Radeon HD 7870 | Q1 2012 | 175 | 2048 | 153.6 |
| nVIDIA GeForce GTX 580 | Q4 2010 | 244 | 1536 | 192.3 |
| nVIDIA GeForce GTX 680 | Q1 2012 | 195 | 2048 | 192.3 |
| nVIDIA Tesla K20 | Q4 2012 | 225 | 5120[b] | 208.0 |
| nVIDIA Tesla C2050 | Q4 2009 | 238 | 3072[b] | 144.0 |
| nVIDIA Tesla C1060 | Q2 2008 | 188 | 4096 | 102.4 |

[a] Maximum device to device bandwidth.
[b] ECC memory supported.

**Table 3.4:** Some other relevant data to take into account for each tested GPU. In the column *Launch* the letter *Q* stands for *Quarter*. The TDP is the Thermal Design Power which indicates the maximum dissipative power of the cooling system: this is taken as an estimate of the GPU power consumption at full load.

Looking at Tab. 3.4 it is worth noting that both the GPU Tesla C2050, and the most recent Tesla K20, support ECC (Error Correcting Code) memory which can detect and correct the most common memory errors ensuring, probably, an improved system stability and more reliable results when running very long simulations. Moreover, Tesla cards have, in general, more on-board memory, up to 5 GB (in some cases 6 GB) in the Tesla K10/K20/K20X. Moreover, some recent CUDA utilities are available only for Tesla cards. However, ECC memory, improved 64-bit performance, large on-board memory and dedicated drivers are surely important characteristics for scientific users, but these features have an important cost too. Moreover, basing our discussion only on declared performance and manufactured characteristics, Radeon GPUs seem to represent a good compromise between 32/64 bit performance, cost and power consumption. Another feature which emerges from Tab. 3.3 is that the recent boards have, in general, a greater number of cores with lower operating frequency than older GPUs. This new "extreme" parallel approach, if combined with a better double precision capability,

ensures higher theoretical performance both in 32-bit and 64-bit precision, at least in those regimes where the GPU is fully "loaded". However, these technical considerations are purely ideal. Actually, the effective measured performance depends on the combination of hardware, software, drivers, characteristics of the motherboard and many other factors that cannot be taken into account in an easy way.

In our case, the benchmarks were performed on one of our workstations at the Department of Physics of "Sapienza", University of Roma. The main characteristics of this workstation (named `astroc12`) and the software used to perform our astrophysical test systems are summarized in Tab. 3.5.[6]

### 3.5.2 Performance measurements

As we have already discussed in previous chapters, `HiGPUs` is such that if we call, hereafter, $m$ the number of particles to update in one integration step, the computational complexity of the $N$-body problem is reduced from $O(N^2)$ to $O(mN)$ where $m$ gets equal to $N$ at the end of the time synchronization process. Here we show a more detailed version of the table 3.1 used in this chapter to study the GPU performance for sections 2, 3, 4, 5, 6, 7, 10 and 11, measuring the time to complete each of them deducing the speed in `Gflops`. In broad lines, our strategy to measure performance can be summarized by the following statement: if, for the $k$-th section, the total number of running GPU threads is $T_k$, the counted floating point operations are $F_k$ and the time to complete the section, in seconds, is $\Delta t_k$, the performance $R_k$ in `Gflops` is obtained by the formula

$$R_k = \frac{T_k F_k}{10^9 \Delta t_k}. \tag{3.17}$$

[6]The Tesla K20 card has been tested thanks to two remote accesses kindly provided by Simon Portegies Zwart, to a machine sited at the Department of Astronomy of the Leiden University (NL), and by the E4 computer engineering to one of their test workstations.

| Astroc12 workstation characteristics | |
|---|---|
| Motherboard | ASUS P6T7 WS SuperComputer |
| Power Supply | Enermax ERV1250EGT 1250 W |
| CPU | 1 Intel core i7 950 @ 3.07 GHz |
| RAM memory | 6 GB (3 x 2GB) 1333 MHz |
| Operating System | Ubuntu Lucid 10.04.2 64-bit version [a] |
| OpenCL | AMD and nVIDIA implementations version 1.2 [b,c] |
| CUDA | version 4.0, May 2011 [c] |
| AMD Drivers | Catalyst 12.6 Linux x86_64 [b] |
| nVIDIA Drivers | 295.75 Linux x86_64 [c] |
| Compiler | gcc/g++ version 4.4.4 |
| MPI | OpenMPI version 1.5.4 [d] |
| GPU | see Tab. 3.3 |
| Software used | `HiGPUs` (direct $N$-body code) |

[a] `http://www.ubuntu.com/`

[b] `http://developer.amd.com/zones/OpenCLZone/Pages/default.aspx`

[c] `http://developer.nvidia.com/category/zone/cuda-zone`

[d] `http://www.open-mpi.org/`

**Table 3.5:** The main characteristics of our workstation used to benchmark the GPUs listed in Tab. 3.3 and Tab. 3.4

To count the floating point operations we refer to Tab. 3.2. In Tab. 3.6, sections 2, 7 and 10 involve memory transfers between the GPU and the CPU through the PCI Express interface. Table 3.6 shows also the total amount of data, in bytes, that must be exchanged. On the other hand, sections 6 and 11 involve only read and write operations inside the GPU on-board memory. This is why, for these sections, we measured the execution times in seconds and we give an estimate of the device-to-device memory bandwidth exploited. Table 3.6 lists, also, the number of bytes that each GPU thread must read (BR)/write (BW) from/to the GPU memory.

### 3.5.3 Astrophysical models

The astrophysical models chosen for our tests include low$-N$ cases (256 stars) up to high$-N$ systems (262,144 stars) and their main parameters are listed in Tab. 3.7. The first three models refer to systems containing bodies randomly distributed in a sphere of unitary radius. The values of $N$ are 256, 512 and 1,024 starting from an initial "cold" condition, i.e. the case where the virial ratio (see chapter 1) is equal to zero. For the masses of the stars, we assumed a bimodal distribution containing $N/2$ "light" particles of mass $m_l$ and $N/2$ "heavy" particles of mass $m_h$. We also considered the presence of an external force-field by mean of a time-independent Plummer potential [82]

$$\phi(r) = \frac{GM_G}{\sqrt{r^2 + b^2}}, \tag{3.18}$$

where $r$ is the distance to the system barycentre, $b$ is a scale radius and $M_g$ is the total gas mass. In the hypothesis that the external potential mimics the role of a gas residual after star formation, the value of $M_g$

| Section | Description | Data for measuring performance |
|---|---|---|
| 1 | Each node determines the stars to be updated and their indexes are stored in an array named *next* containing $m$ elements | Not used[a] |
| 2 | Each node copies to its GPUs the array containing indexes of $m$ particles | $4m$ `Bytes` |
| 3 | If $k$ is the number of GPUs that will be used in the numerical integration, the predictor step of $N/k$ stars is executed | `81 ops (DP)` |
| 4 | Each node computes the forces (and derivatives) of $m$ particles due to $N/k$ bodies | `SP : 82`<br>`15 ops (DP)` |
| 5 | Each node reduces the calculated forces and derivatives of *Bfactor* blocks | `10 ops (DP)` |
| 6 | Each node adjusts conveniently the reduced values | `32 BR`<br>`32 BW` |
| 7 | The CPUs receive the accelerations from the GPUs | $96m$ `Bytes` |
| 8 | The `MPI_Allreduce()` functions collect and reduce accelerations from all the computational nodes | Not used[a] |
| 9[b] | Corrector step and time step update for $m$ stars | Not used[a] |
| 10[c] | The reduced accelerations and the corrected positions and velocities of $m$ bodies are passed to the GPUs of each node | $192m$ `Bytes` |
| 11[c] | The GPUs rearrange the updated particles following the original indexes stored in the array *next* | `36 BR`<br>`24 BW` |

[a] `This section involves only the CPU.`
[b] `This section has been ported on GPU in the latest version of HiGPUs.`
[c] `This section is not needed if the corrector step is performed on the GPU.`

**Table 3.6:** The main sections of our code performed for each time step.

is determined by assuming a value for the Star Formation Efficiency, defined by

$$\epsilon = \frac{M_*}{M_* + M_g}, \tag{3.19}$$

where $M_*$ is the total mass in stars. Here we take $\epsilon = 0.3$ as a likely astrophysical value. On this basis we define three simple reference models, indicated with the symbols `V1`, `V2` and `V3`, to (roughly) mimic the initial state of young and very young open clusters which are observed in sub-virial conditions, mass segregated, despite their age, and still embedded in their native gas. A preliminary scientific analysis of the results obtained from these simulations will be presented in chapter 6 while here we limit the analysis to the GPUs performance. We also sampled the initial conditions for other $N$-body systems [7]. from two King models [56], indicated in Tab. 3.7 with `K1` and `K2`, with $N_{K1} = 65,536$, $N_{K2} = 32,768$. For the King models we assumed two values for the dimensionless central concentration parameter, $W_{0_{K1}} = 7$ and $W_{0_{K2}} = 9$. In model `K1` an Initial Mass Function, like that described in [59], has been adopted while in the model `K2` all the stars have the same mass, $m = \frac{1}{N}$. We also sampled a King model, indicated with the letter `K3`, with $N_{K3} = 262,144$ stars, $W_{0_{K1}} = 6$ and the same mass function used for the model `K1`, embedded in a rough representation of the Milky Way bulge potential as a Plummer analytical potential [8] and moving on a circular orbit at 2 kpc from the centre of the system barycentre. We also sampled a Plummer model, listed as `P1`, having $N_{P1} = 16,384$. In the Plummer model all the stars have the same numerical mass $m = \frac{1}{N}$.

All these models were generated using `McLuster` [60] and all the mentioned test cases have been followed up to 10 time units, which is an extension in time sufficient to obtain a reliable averaged performance for all the different sections of our $N$-body code. Specifically, we developed also a code which can generate $N$-body models, just like `McLuster`,

[7] The detailed description on how to sample initial (stable) conditions for stellar systems is presented in chapter 4

starting form a generic density profile in spherical symmetry. The details can be found in chapter 4.

### 3.5.4 Performance results

#### Evaluation of the mutual forces

First of all we analyse the most important section of any $N$-body code: the evaluation of the accelerations and, for the Hermite's 6th order scheme, some of their time derivatives. For populous stellar systems this is, by far, the section which takes most of the execution time, therefore the performance exhibited in this part is of crucial importance for realistic scientific applications. On the other hand, for small and intermediate$-N$ systems ($N \lesssim 16k$), as we will see later, the time spent to execute this evaluation step becomes comparable to (or even smaller than) that spent to complete other `HiGPUs` sections. This underlines the importance to have powerful and efficient hardware on both small and large scales. Nevertheless, to know the real performance and gain of a specific GPU on the overall evolution (10 time units for this work) the figures shown in this section must be integrated with the histograms in Fig. 3.20, 3.21, and 3.22 that represent the measured wall-clock times for each tested GPU to evolve each test system.

#### large N case: systems K3 and K1

In Fig. 3.11 we show the speed performance of the various GPUs in the execution of the evaluation step of `HiGPUs` , in function of the number $m$ of particles to be updated, in a generic time step. We refer to the system `K3` only because the resulting plot for system `K1` does not point out significant differences. Fig. 3.11 shows that, in the whole range of values of $m$, the Radeon card HD7970 performs over 1 `Tflops` while the other

| Model | Notation | N | System parameters | Background parameters |
|---|---|---|---|---|
| Homogeneous sphere + Plummer background | V1 | 256 | $R = 1$ $M = 0.3$ | $b = 1$ $M_g = 0.7$ |
| Homogeneous sphere + Plummer background | V2 | 512 | $R = 1$ $M = 0.3$ | $b = 1$ $M_g = 0.7$ |
| Homogeneous sphere + Plummer background | V3 | 1,024 | $R = 1$ $M = 0.3$ | $b = 1.0$ $M_g = 0.7$ |
| Plummer sphere | P1 | 16,384 | $b = 1$ $M = 5$ | no backgorund |
| King distribution | K2 | 32,768 | $W_0 = 9$ $r_c = 0.2$ $M = 5$ | no background |
| King distribution | K1 | 65,536 | $W_0 = 7$ $r_c = 0.2$ $M = 5$ | no background |
| King distrib. in a Plummer background | K3 | 262,144 | $W_0 = 6$ $r_c = 0.01$ $M = 0.001$ | $b = 4$ $M_g = 14$ |

**Table 3.7:** The complete set of simulations performed for our benchmarks. $R$ and $M$ represent, respectively, radius and mass of the stellar system. The parameter $b$ is the Plummer's core radius (see Eq. 3.18), $M_g$ is the total mass of the analytic, stationary background, if present, $r_c$ is the King's core radius and $W_0$ is the dimensionless central concentration [56]. All the simulations are performed in units such that $G = 1$, while the length and mass units are chosen for computational convenience as in column 4.

GPUs show a speed from 10% (Tesla C1060) to 50% (Radeon HD6970) up to 75% (Tesla K20) that of the HD7970 board. Tesla C2050 and HD7870 have approximatively the same performance while GTX cards do not get considerable results mainly because of the low double precision computing power (see Tab. 3.3). Moreover, the recent GTX 680 (Kepler Architecture), has a speed performance a factor 1.4 worse than that of the previous generation GTX 580 (Fermi Architecture). This is mainly due to the ratio of performance in 64 bit precision operations between these two cards. Nevertheless, it is curious to highlight that, although the technical features of the GTX 680 and HD7870 are very similar, the performance of the HD7870 is, in this large-$N$ regime, about a factor 1.6 higher of GTX 680. This is likely due to that an HD7870 can run up to 51,200 GPU threads in parallel while a GTX 680 only up to 16,384. Therefore, the high parallel capability of the HD7870 is clearly preferable in regimes of full load state of the GPU (as happens in the large-$N$ case). Despite tuned and different optimizations introduced in our code working when $m$ is smaller than the maximum number of parallel threads that a GPU can run simultaneously, we can see a slight decay of performance when $m \lesssim 400$ for system `K3` (700 for system `K1`) especially in the case of the Radeon HD7970, HD6970 and Tesla K20. This is not surprising because these three cards are massively parallel. These GPUs have a large number of processing elements with low clock frequencies and an HD7970 can run up to 81,920 threads simultaneously while an HD6970, as well as a Tesla K20, up to 32,768. Therefore, it is difficult to load completely these GPUs in the above low $m$ regime while the others GPUs are easier to exploit having, in general, both less resources available and less theoretical computing power. This explains why the performance decay at low $m$ of these latter GPUs is almost negligible. In any case, we can affirm that, for a direct $N$-body code and, more generally, for a kernel which fully loads the GPU using both single and double precision operations, an HD7970, an HD6970 or a Tesla K20 represent the best choice to obtain scientific results in a short time.

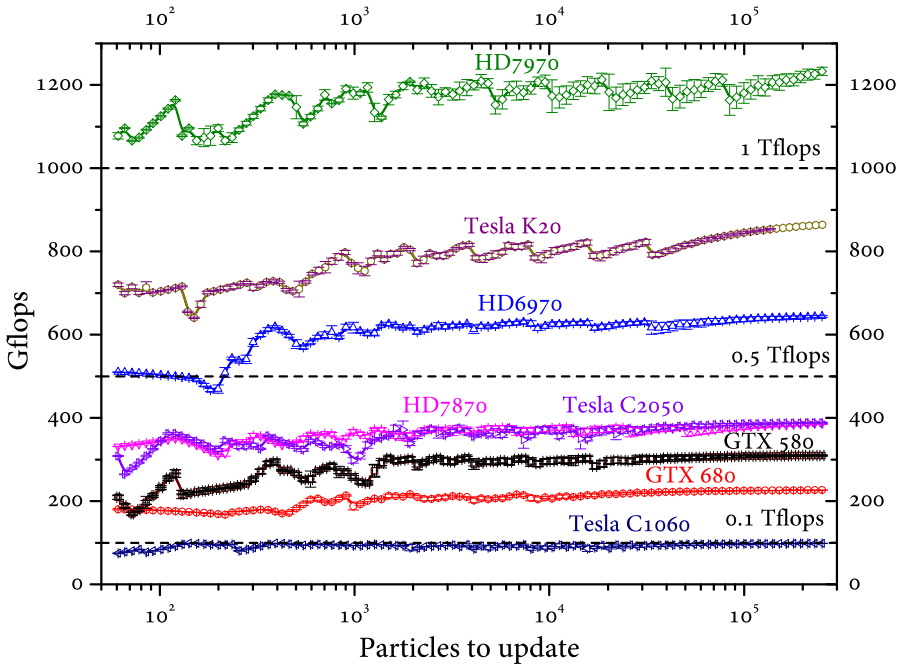


**Figure 3.11:** Speed performance of the tested GPUs, in `Gflops`, as a function of the number of particles to update. This figure refers to system K3, with $N$=262,144.

### intermediate-N case: systems P1 and K2

In Fig. 3.12 we show the performance of the tested GPUs in the same frame adopted for Fig. 3.11. This figure refers to the system `P1`, chosen as reference case for this regime of intermediate $N$. Radeon cards, also in this regime, exhibit higher performance than the other GPUs with reference to the evaluation step of `HiGPUs` . The performance of the Tesla K20 remains always between the two Radeon GPUs, except for low values of $m$ ($m \lesssim 200$) in which it performs slightly better. The gain of the massively parallel cards is relevant when $m \gtrsim 500$ while for smaller values of $m$ the performance decay is more evident than in the previous high-$N$ case for all the GPUs although the Tesla C1060 remains in a state of full load (around 80 `Gflops`) because it has, both, less cores and much lower theoretical performance than the other cards.

We found that system P1 ($N$=16,384) is a lower limit for the number of particles per GPU in the sense that below this $N$ the time spent by, for example, an HD7970, to complete the other sections of our code becomes significant if compared to the total execution time. To remark more this idea, Fig. 3.13 shows the ratio between the sum of the times spent by an HD7970 to complete all the other parts of HiGPUs and that to complete just the evaluation step, as a function of the number of particles to update for our test systems. The fraction of the time spent to evaluate accelerations to the total execution time is about 65% for system P1 using an HD7970. The remaining 35% is equally divided in memory transfers and reduction of partial forces. Therefore, while for systems K1 and K3 the evaluation step is, by far, the most important part, this is no longer true for the other tested systems. Fig. 3.13 is useful to show that for systems with $N \lesssim 16k$ the overall hardware performance is determined also by the other sections of HiGPUs .

### Low-N cases: systems V1, V2 and V3

Even if, in this regime, one may not need to use powerful computing accelerators because the total execution time is well below that spent to integrate systems in the intermediate and large-$N$ cases, it is very interesting to study how GPUs perform when they are not totally loaded. This may also give us some general and useful information in the case when more than one computing node is available. In fact, for example, a system of $N = 1,024$ bodies on a single GPU can be considered (almost) equivalent to a system composed by $N = 1M$ bodies distributed over 1,024 GPUs. Therefore, considering the low-$N$ regime, we can argue some considerations about the performance that would be got running large-$N$ systems over a set of GPUs.

As we said above, and as it is shown in Fig. 3.13, in this regime it is important to consider how the GPUs perform not only in the evaluation

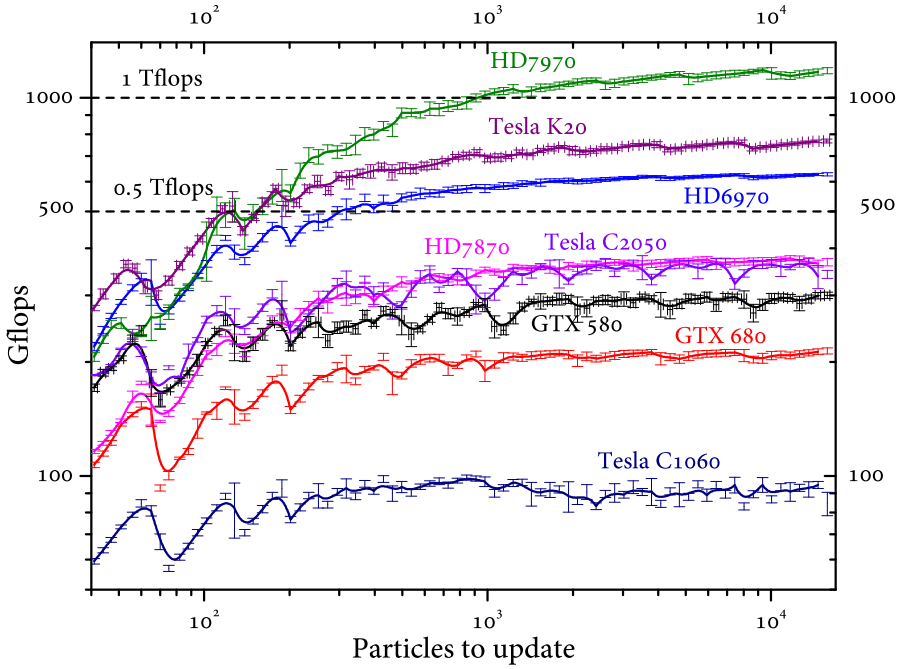


**Figure 3.12:** Speed performance of the tested GPUs, in `Gflops`, as a function of the number of particles to update. This figure refers to system P1, with $N = 16{,}384$.

step but also in other sections of `HiGPUs` . Before discussing this, let us examine the performance in the evaluation step. As we can see, for example, in Fig. 3.14, relative to system `V1`, the situation is completely changed with respect to large-$N$ systems. The performance of GTX 580 and Tesla C2050 become comparable even if they remain well below their maximum peak. On the other hand, the Radeon cards, the Tesla K20 and the old generation Tesla C1060 are slower. The growth of performance is, on average, linear for all the GPUs because they are far to be fully loaded and the performance increases with the number of running threads. In general, the larger the distance from the full-load state, the closer to the linear speed increase. This trend is particularly evident for Radeon GPUs and Tesla K20 and less for other nVIDIA cards, whose linear performance growth disappears completely already for system `V3`, (see Fig. 3.15). In fact, in system `V3`, GTX 580, GTX 680 and Tesla

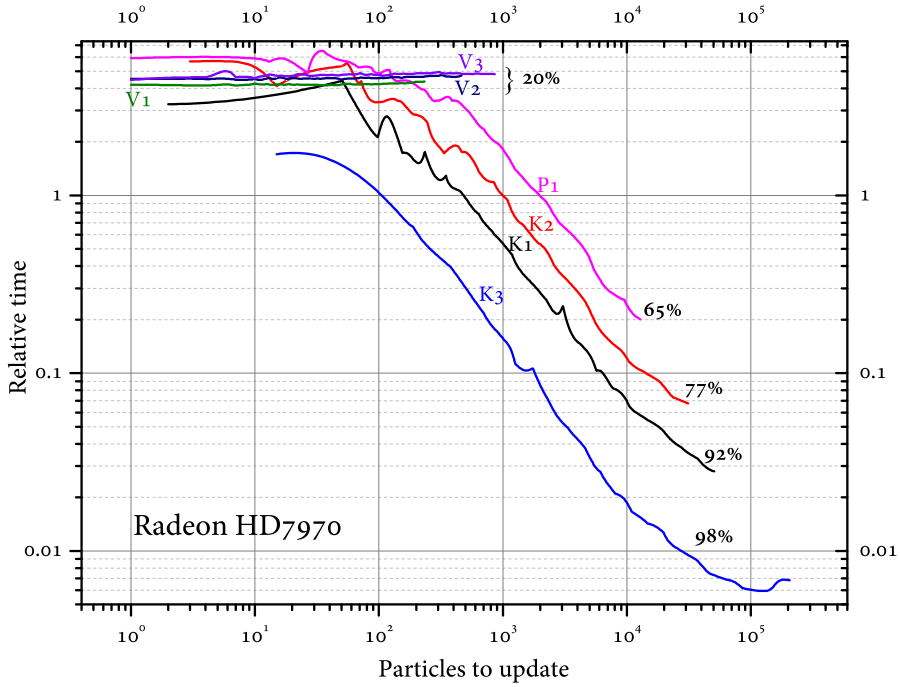


**Figure 3.13:** Relative importance of all the code sections (excluding the evaluation) to the evaluation section in function of the number of particles to update in different cases. The various curves are labelled by the percentage time spent in the evaluation.

C1060 get closer to their measured performance peak while Radeon GPUs and Tesla K20 maintain their approximatively linear trend being still very distant from their full load state. The Tesla C2050 performance can further increase a little although the growth is no longer linear. We do not report further figures for the regime in which $N \in [1,024; 16,384]$ because the evolution of the speed performance of the tested GPUs can be naturally argued from what has been already shown and discussed. Actually, this is a transition phase in which the situation continues to evolve and, in particular, for $N = 4,096$ Radeon GPUs and Tesla K20 have already exceeded the performance of other nVIDIA GPUs and the results become very similar to that showed in Fig. 3.12. At the light of this analysis one concludes that a GTX or a Tesla C2050 card could be

the right choice to perform direct $N$-body simulations in this regime but we need to consider also other factors that will be taken into account in the next section.

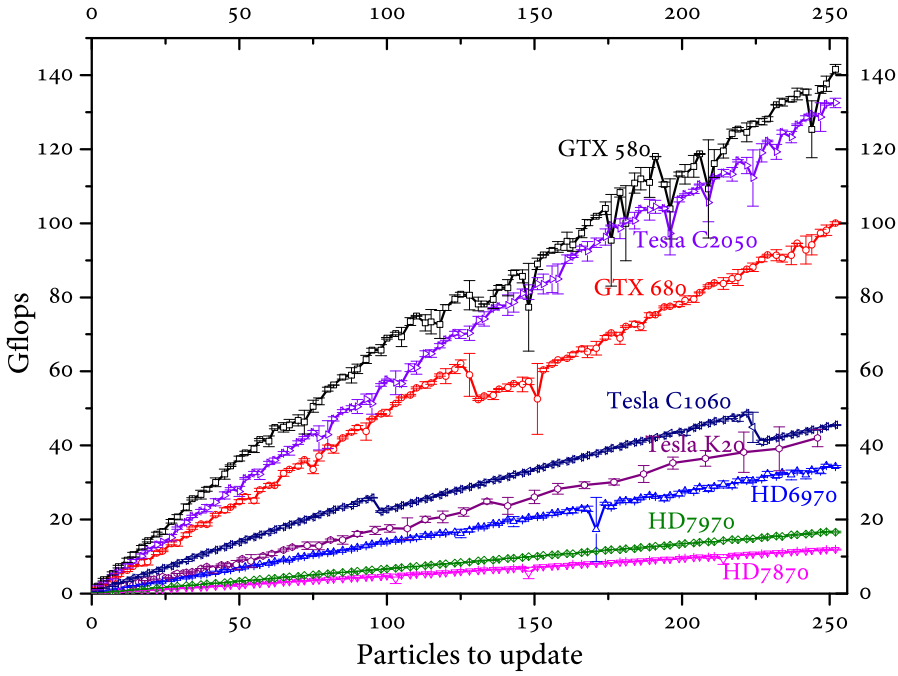


**Figure 3.14:** Speed performance of the tested GPUs, in `Gflops`, as a function of the number of particles to update. This figure refers to system V1, with $N=256$.

## 3.5.5 Other important code sections

As seen in Fig. 3.13, while the evaluation section constitutes the most important part for large-$N$ systems, in the case of small-$N$ we must consider also the performance obtained in other sections, which we divide, for convenience and clarity, into 3 groups (see also Tab. 3.6)

1. Host-to-Device and Device-to-Host transfers (sections 2, 7 and 10);

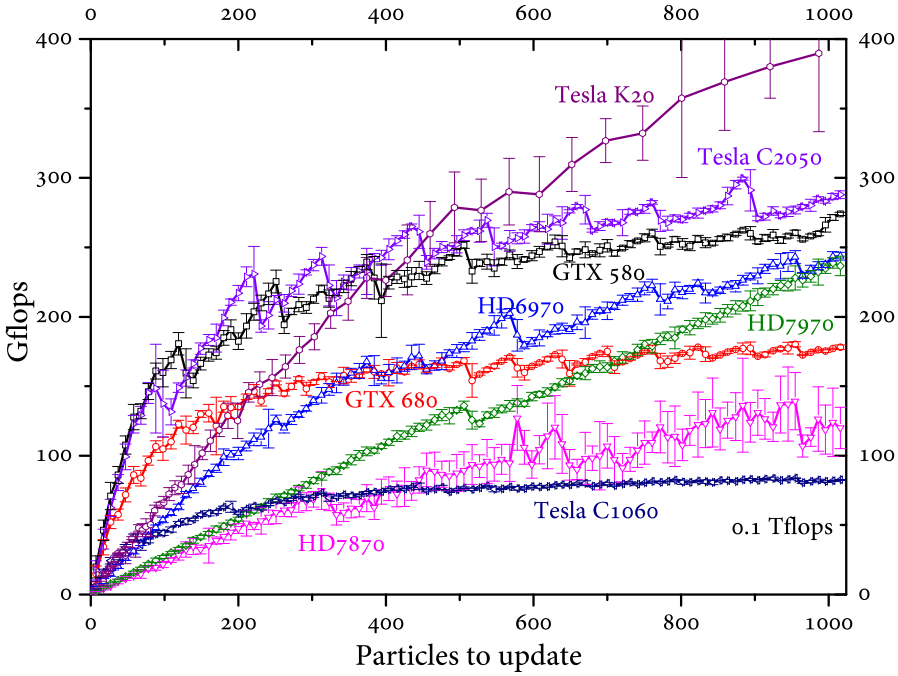


**Figure 3.15:** Speed performance of the tested GPUs, in `Gflops`, as a function of the number of particles to update. This figure refers to system V3, with $N = 1{,}024$.

2. Reduction of partial forces (section 5);

3. Device-to-Device transfers (sections 6 and 11).

It is worth noting, again, that the more recent improvements introduced in `HiGPUs` are such that there is the possibility to run the correction step directly on the GPU. This improves performance for large$-N$ systems (especially if we run `HiGPUs` on more than one computing node) and, in addition, Sections 6 and 11 of our code are not needed anymore. Moreover, the prediction step is not considered here being always below the other sections in terms of computing time. Nevertheless, to develop this work we used an older version of `HiGPUs` whose corrector was performed on the CPU, and, in this case, the above listed three groups of sections contribute, with good approximation, for about 1/3 each to the

execution time not spent in the evaluation of the forces. Let us examine the performance exploited in these 3 sections.

### Host-to-Device and Device-to-Host Bandwidth

Fig. 3.16 shows the resulting bandwidth, normalized to that of Tesla C1060, in function of the amount of data transferred. The curves are obtained by an arithmetic average of the performance measured for sections 2, 7 and 10 because no significant differences were found transferring data from/to the host and device. We do not show in Fig. 3.16 the results obtained for the Tesla K20 because we noticed that its bandwidth has a peculiar behaviour which it has been reported, for more clarity, separately in Fig. 3.17. As it can be seen in Fig. 3.16, the results for the GTX 580 and GTX 680 are almost identical. We have also indicated, with vertical dashed lines, the maximum data transfer during the dynamical evolution of our test systems. It can be seen that the bandwidth of the Radeon GPUs is constantly below the bandwidth of nVIDIA GPUs. The reason is not easily determined but, surely, the drivers play an important role. What is important for our scopes is that this performance deficit is critical for systems `V1`, `V2` and `V3` in which data transfers between the host and the device become one of the bottlenecks for our simulations. Actually, for very low-$N$ systems, Radeon GPUs loose about a factor 3.5 in performance almost independently of the number of particles in a block. This degradation of performance adds to what is lost in the evaluation step in these regimes (see Fig. 3.14 and Fig. 3.15 ). Therefore, the GTX 580/680 and the Tesla C2050 perform better also on memory transfers between host and device so they are a very good choice in regimes of weak load. Anyway the situation of weak load, i.e. low-$N$, is not in many cases critical on a computational side. Radeon GPUs improve performance when the amount of data to exchange is large enough (greater than 100 MB) but, at this level of amount of

data transfer, the differences of bandwidth performance among different GPUs are definitively negligible.

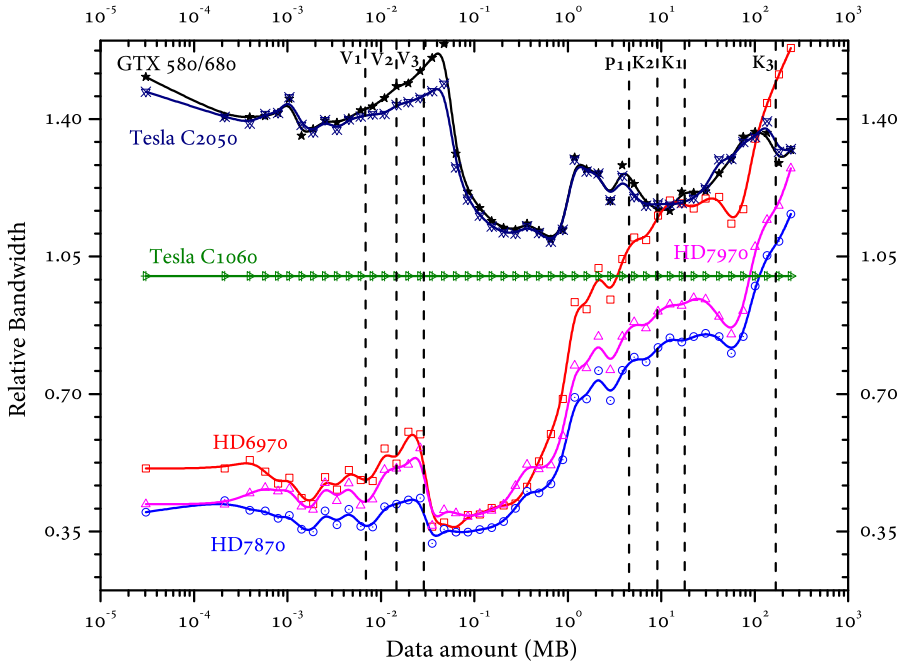


**Figure 3.16:** Bandwidth, normalized to Tesla C1060, of the tested GPUs as a function of the amount of data to exchange (in MB). This figure gives also as straight vertical lines the upper limit to the amount of data that are transferred for each of the test systems.

## Reduction of partial forces

The optimizations introduced in our code are based mainly on the determination of the maximum number of threads that the GPU can handle at the same time. `HiGPUs` automatically calculates this number, $P_t$, and, if $m \lesssim P_t$, the standard one-to-one correspondence between particles to update and parallel threads is increased in order to exploit, as much as possible, all the capabilities of the GPU. In this case, the correspondence is increased to $1{:}k$ ($k > 1$), which means that each thread calculates the

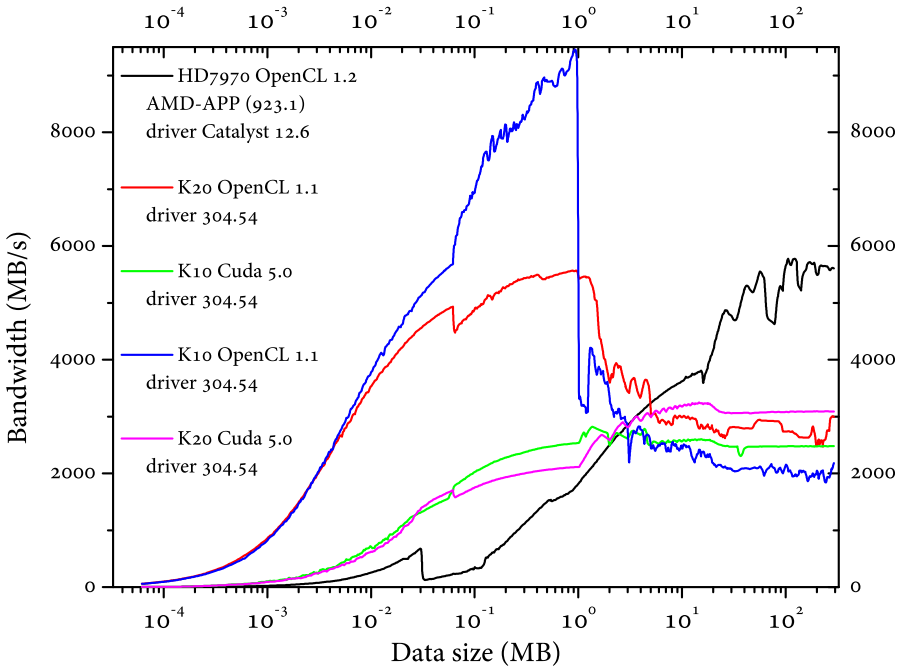


**Figure 3.17:** Bandwidth of some of the different GPUs examined in this paper, in function of the amount of data transfer. In the figure we label the various GPUs with the operational software and driver version used. Note the somewhat steep decline of the bandwidth of the Tesla Kepler nVIDIA cards when using OpenCL at 1MB, while the same cards using CUDA flatten at data transfer amount above 10 MB at 3 GB/s level (that is about 50% of the HD7970 bandwidth).

force acting on its own particle due to $N/k$ bodies. The performance of the evaluation step can be improved up to a factor 100 using this strategy (see Fig. 3.9). Nevertheless, in this way we introduce another operation which is the reduction of $mk$ forces, all of them stored as double precision (64-bit) values. The latter operation becomes important for the small and very small-$N$ systems `V1`, `V2` and `V3`, therefore in Fig. 3.18 we show the performance of the tested GPUs in reducing partial forces for $m < 1,024$ which is the typical regime in which the above described approximation strategy is active and relevant in terms of execution time. As usual we normalize the result to one GPU (in this case

we use the Tesla C1060 as reference). Similar to what previously seen, the GTX and Tesla C2050 cards perform better than Radeon and K20 cards that loose a factor $> 4$ with respect, for example, to a GTX 580. We may say that, in general, GTX and Tesla C2050 GPUs are better exploited and maintain high efficiency on both small and large scale problems while the same cannot be said for Radeon GPUs. In fact, at these regimes of both weak load and arithmetic intensities, the single-core working frequency and lower latencies accessing GPU memory become discriminant for better and worse performance. It would be interesting to have a sort of boost of the GPU single-core frequency which should be active whenever the GPU is recognized to be not in a full-load state. This could guarantee a massively parallel GPU which would remain very efficient (like the GTX 580 for example) even for weak-load regimes.

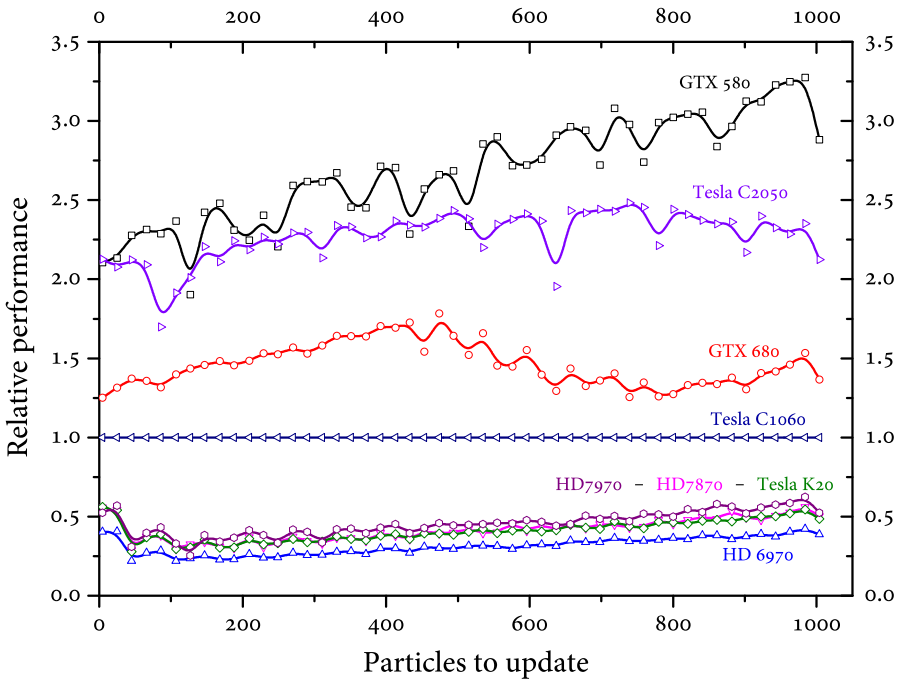


**Figure 3.18:** Performance in executing the reduction of partial forces, normalized to Tesla C1060, of the tested GPUs, as a function of the particles to be updated

### Device-to-Device bandwidth

For small-$N$ systems, another important section in terms of the total execution time is that involving exchanges of data inside the global memory of the single GPU. There are two kernels in `HiGPUs` which perform this sort of Device-to-Device operations, and we measured performance of these sections in terms of GB transferred per second, considering the values listed in Tab. 3.6. In Fig. 3.19 we show the results normalized, for convenience, to the performance of the GTX 680. Once again the GTX GPUs and the Tesla C2050 are well above the Radeon GPUs, at least for $m \lesssim 10^4$. The old generation Tesla C1060 card looses a factor between 1 and 2.5 respect to the GTX 680. Radeon GPUs and Tesla K20 reach performance of the other nVIDIA cards only for $m > 10^5$; Tesla C1060 is limited by its low theoretical device-to-device bandwidth (see Tab. 3.4). Anyway, in this large $m$ regime, the difference in performance executing memory transfer operations is negligible with respect to the total execution time.

## 3.6 A possible application: the Milky Way Nuclear Star Cluster

We briefly show in this Section the total execution times needed to evolve our astrophysical systems over 10 time units using the GPUs under test. Each system has been integrated using a proper softening parameter, $\epsilon$, in the pair-wise force. For systems `V1`, `V2` and `V3`, $\epsilon \simeq 3 \cdot 10^{-4} \langle D \rangle$ where $\langle D \rangle$ is an estimate of the nearest neighbour distance i.e.

$$\langle D \rangle = \frac{R}{\sqrt[3]{N}}. \tag{3.20}$$

For systems `K1`, `K2` and `K3` we used $\epsilon \simeq 10^{-2} r_c$ and, for system `P1`, $\epsilon \simeq 4 \cdot 10^{-3} b$. The results are shown in Fig. 3.20, 3.21 and 3.22 in

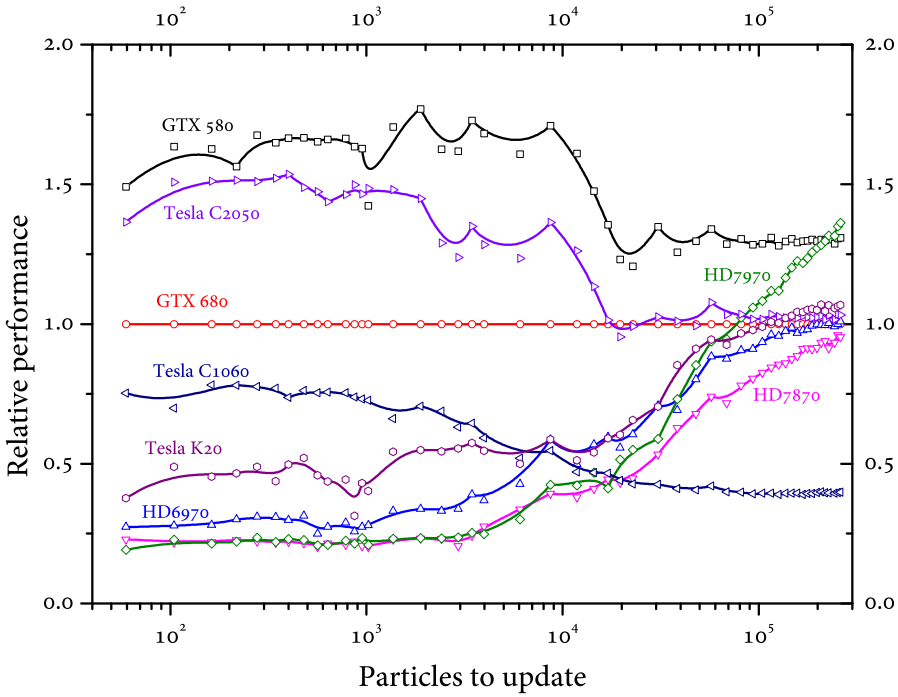


**Figure 3.19:** Performance in executing device-to-device transfers, normalized to GTX 680 for convenience, of the tested GPUs, as a function of the particles to be updated

the form of histograms in which the wall-clock times have been normalized to those of HD7870, for convenience. As an example, the integration of the system K3 for $10^9$ years will require around 1,920 days using an HD7870 and only around 600 using a single HD7970. Specifically, using our very small, green and cheap cluster composed by two computational nodes each composed by two multicore CPUs and 4 HD 7970 GPUs, we may evolve the system K3 for $10^9$ years in around 75 days of simulation, reaching a peak of 10 Tflops+ of sustained performance. For the sake of future applications of actual astrophysical interest we are dealing with the formation and the long term (Gyr) evolution of dense stellar systems around very compact and massive objects, like black holes. Such systems are often observed in the central regions of galaxies; in particular, more steps forward have to be done in the numerical simulations of the so called Milky Way Nuclear Star Cluster, whose

model of formation and evolution are still under debate (see [12] and [30]). Through preliminary tests, we estimated that we can evolve this system, modelled using $N = 2M$ stars plus a central massive black hole, up to 1 Myr in around 8 hours. (That is 8,000 hours to evolve this system up to $10^9$ years). Although following a long term evolution is not possible using only eight HD7970, it can be done with our code on large hybrid supercomputers in the world (especially Titan, which is composed by 18,688 nVIDIA Tesla K20X). If we suppose, as we saw in our benchmarks, that the performance of a single K20X is comparable with that of one HD7970, the availability of 256 GPUs (less then 2% of Titan [94]), will allow us to finish the mentioned simulation in $\sim 1$ month, reaching an unprecedented spatial resolution at a sustained speed around 0.3 `Pflops`, which is, definitely, a very good result for large-$N$ direct simulations.

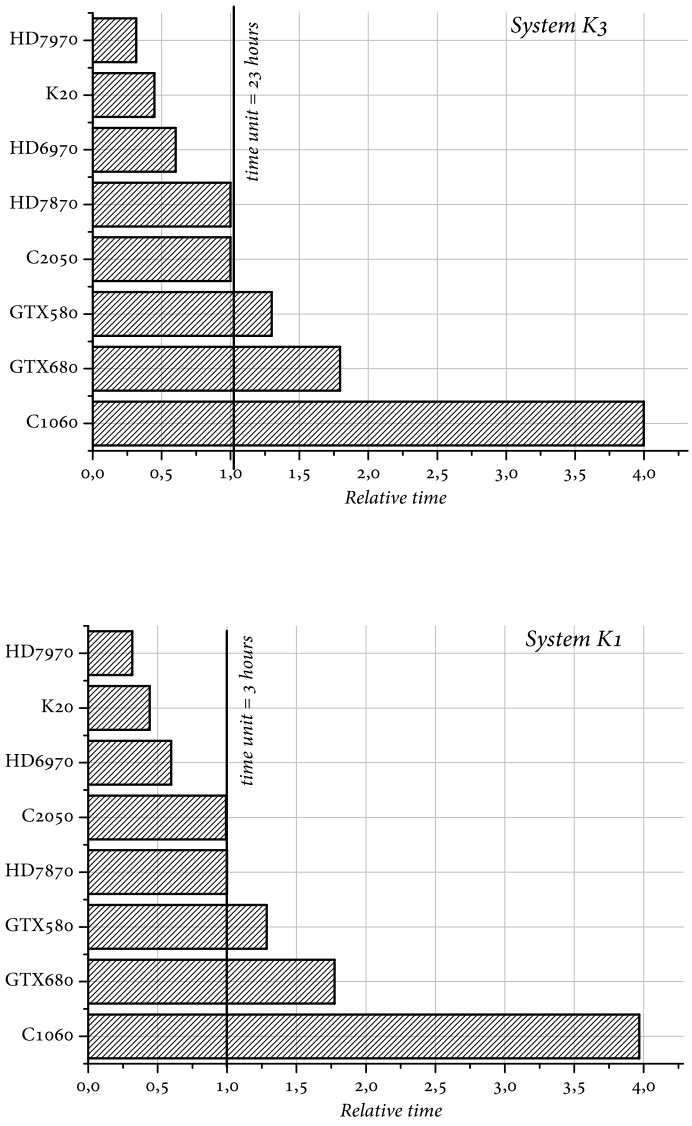


**Figure 3.20:** The execution times needed to evolve systems `K3` and `K1` over 10 time units using different GPUs. The performance are normalized to the HD 7870. The time unit is reported in each figure.

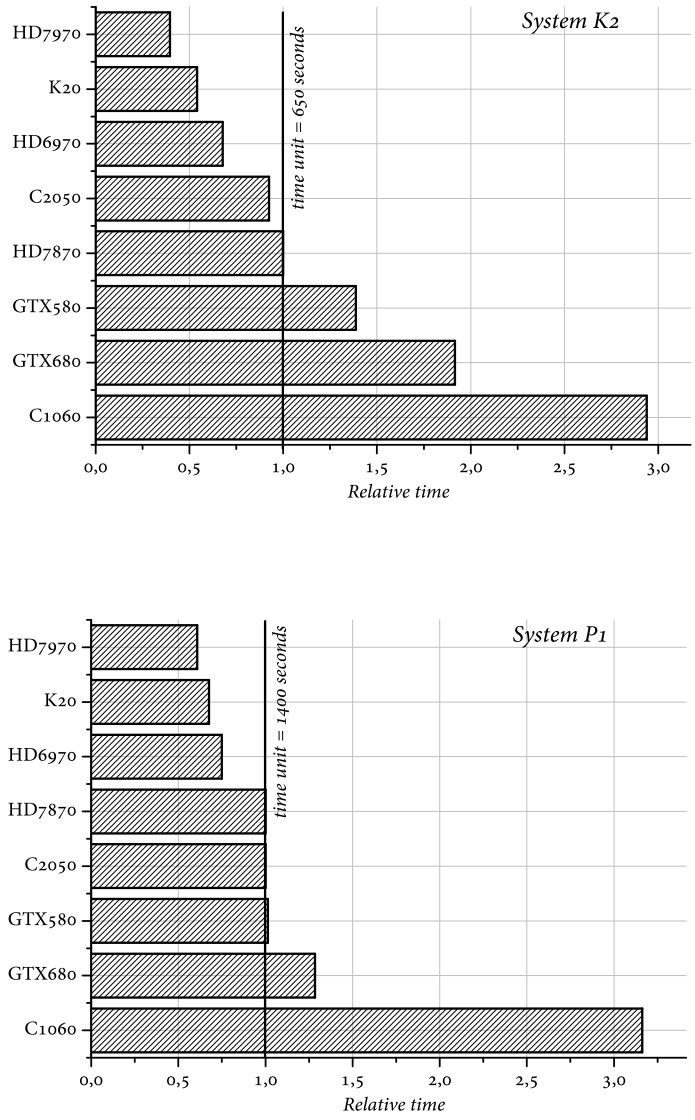


**Figure 3.21:** The execution times needed to evolve systems `K2` and `P1` over 10 time units using different GPUs. The performance are normalized to the HD 7870. The time unit is reported in each figure.

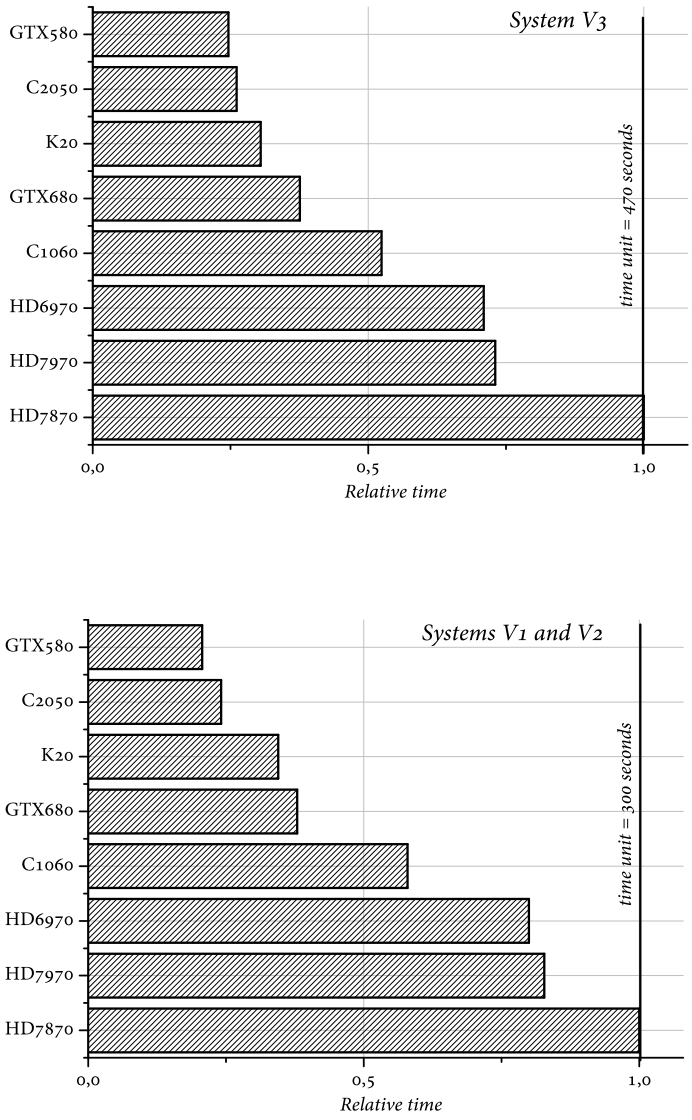


**Figure 3.22:** The execution times needed to evolve systems `V1`, `V2` and `V3` over 10 time units using different GPUs. The performance are normalized to the HD 7870. The time unit is reported in each figure.

## 3.7 Final remarks and future developments

In this section we compared performance of some Graphic Processing Units produced by different firms when applied to a scientific application. As test topic we chose the integration of the motion of $N$ objects interacting via the pairwise Newtonian gravitational force, and as code to do this one that evaluates these forces via direct summation and performs the integration in time by mean of a Hermite's 6th order method. Regarding the accuracy of `HiGPUs` , we note that the relative error in total energy and angular momentum has always kept below $5.0 \cdot 10^{-9}$ except for computationally critical runs (`V1`, `V2` and `V3`) in which it was below $1.0 \cdot 10^{-3}$.

To allow a compared benchmark on GPUs of different makes a portable version of the code is needed. In fact, the GPUs produced by nVIDIA are conveniently programmed in Compute Unified Device Architecture (CUDA) while other vendors' GPUs do not support this paradigm and need to be programmed in OpenCL which, although "young" and thus not as developed as CUDA and still awkward to use, shows a very good efficiency. Nevertheless, we note that very few scientific applications have been implemented in OpenCL, so far. We are aware of another work by Hauschildt and Baron [50]. The performance tests consisted of runs of `HiGPUs` with initial conditions aiming at the representation of the evolution of some stellar systems of astrophysical interest (stellar clusters) in three different ranges of the total number of interacting objects, $N$: low$-N$, intermediate and large$-N$ cases. The sense of low, intermediate and large has to be referred to the O($N^2$) complexity of the high precision computations done by direct summation. The description of the various test cases is given in Table 3.7.

The main result obtained may be summarized as follows:

1. as expected, the global performance is a combination fo the "brute" computational power of the GPU and of the host-to-device and device-to-host bandwidth;

2. the bandwidth exploited by the GPUs examined is higher for nVIDIA cards when there are few data to exchange, while at high levels the AMD GPUs are faster;

3. the breakdown of nVIDIA cards performance at about 1 MB of data transfer is not completely understood. It may be due to the particular version (304.54) of the driver used, at least for Tesla K10 and K20 cards, while it works fine for nVIDIA GTX 580 and 680 when using CUDA (OpenCL does not work properly with this driver version on GTX cards);

4. the highest computing speed is reached, in the majority of the examined cases, by the AMD HD 7970;

5. the AMD HD bandwidth is not as high as that of the whole set of nVIDIA GPUs tested whenever the amount of data to exchange is not over a certain threshold, over which the HD 7970 performs as well as the more expensive Tesla C and Tesla K GPUs.

The previous points imply what we have practically found and tested, i.e. that:

1. the global performance of the AMD HD7970 is the highest of the GPUs examined here whenever it is "load" enough to exploit its intrinsically higher computational power without penalization on the bandwidth side, thing that occurs in all the test cases studied here;

2. the nVIDIA Tesla GPUs of the Fermi and Kepler generations performs in a range of speed from 10% (Tesla C1060) up to 75% (K20);

3. the nVIDIA cards of the GTX family have speed performance in between the Tesla C1060 and those of Tesla C2050 and AMD HD7870, these latter being pretty similar.

At the light of previous considerations and results, we may say that it is absolutely well pursuing code implementations in both CUDA, to exploit at best the performance of the very stable and controlled nVIDIA GPUs of the Fermi and Kepler class, and OpenCL, which is needed to use the high power to price (and power consumption cost) of the GPUs of the AMD HD series make. As expected, some weak points are found in using AMD GPUs, like that of some instability seen when using AMD drivers of different releases. No particular problems rise, on the other side, by the absence in the AMD GPUs of the Error correcting code memory (ECC) available on the Tesla C2050 and K20. The on board memory limited to 3GB may represent, for the AMD HD7970 examined here, a limitation for some scientific applications, although it did not limited its performance in the cases studied in this work. The GPU hardware evolution is fast, and some developments have been announced by the GPUs producers, so no specific firm conclusion and operational suggestion may be reliably drawn to be applied over a reasonable time range. It would be interesting to see how the most recent GPUs (Radeon R9 290X, GeForce Titan and Tesla K40) perform even if the double precision theoretical maximum performance of the R9 290X has been ported from 1/4 (as it was for the HD 7970) to 1/8 that of single precision limiting, therefore, our interest to test it using our code `HiGPUs` . Anyway, at this stage it seems that a good receipt to follow when setting up a hybrid computational platform, especially of small-intermediate size, is to carefully consider the weights to give to the various involved parameters (cost of the single GPUs, stability and robustness of the system,

quality of drivers, etc., bandwidth, power consumption on a side, performance and easiness in programming on another side) when aiming to a specific category of scientific topics.

# 4 The initial conditions of stellar systems

## 4.1 The distribution function $f(\mathbf{x}, \mathbf{v}, t)$

In this chapter we will show how to generate proper initial conditions (positions, velocities and masses) for a generic $N$-body system. Typically they are collected into a computer file which will be able to be used as input of a generic $N$-body code like, for example, our `HiGPUs` in order to study the dynamical evolution of the sampled model. First of all we need to introduce the concept of *distribution function* (hereafter `DF` ) which will be fundamental to assign a proper velocity to the $N$ particles. Given an unit of volume $\mathrm{d}^3\mathbf{x}\mathrm{d}^3\mathbf{v}$ around the position $\mathbf{x}$ and velocity $\mathbf{v}$ in the phase-space, we define the distribution function $f(\mathbf{x}, \mathbf{v}, t)$ as a function such that $f(\mathbf{x}, \mathbf{v}, t)\,\mathrm{d}^3\mathbf{x}\mathrm{d}^3\mathbf{v}$ gives us the probability, at a given time $t$, to find a generic star with position between $\mathbf{x}$ and $\mathbf{x} + \mathrm{d}\mathbf{x}$ and velocity between $\mathbf{v}$ and $\mathbf{v} + \mathrm{d}\mathbf{v}$. Directly from its definition comes its normalization which requires

$$\int f(\mathbf{x}, \mathbf{v}, t)\,\mathrm{d}^3\mathbf{x}\mathrm{d}^3\mathbf{v} = 1\ . \tag{4.1}$$

Because the stars move in the phase-space, the probability to find a generic star in a certain position with a certain velocity changes with time. In collisionless systems[1] the `DF` evolves in such a way that the

[1] A *collisionless system* is a system whose typical dynamical time-scales are significantly smaller than its relaxation time. In particular, in such systems, close encounters between stars did not play an important role to their overall evolution.

probability is conserved that is the DF must satisfy the generalized form of the *continuity equation*

$$\frac{\partial f}{\partial t} + \frac{\partial}{\partial \mathbf{w}} \cdot (f \dot{\mathbf{w}}) = 0 \tag{4.2}$$

where $\mathbf{w} = (\mathbf{x}, \mathbf{v})$ and $\dot{\mathbf{w}} = (\dot{\mathbf{x}}, \dot{\mathbf{v}})$. On the other hand, for collisional systems, that is when close encounters between stars are taken into account, the phase-space probability density of stars, around a given point, changes with time accordingly to the so called *encounter operator* $\Gamma[f]$. Some mathematical considerations about the function $\Gamma[f]$ yield us to write the so called *master equation* which describes the complete evolution in time of the DF in the case of collisional systems. Nevertheless, in this work, we are interested in investigating collisionless systems which will be in equilibrium for arbitrary large interval of times but, for a deeper analysis about this topic, it is possible to consult [18]. In order to obtain an equation for the evolution of the DF for collisionless systems we use the Hamiltonian equations (see eq. (1.95)) to simplify equation (4.2)

$$\begin{aligned}
\frac{\partial f}{\partial t} + \frac{\partial}{\partial \mathbf{x}} \cdot (f \mathbf{x}) + \frac{\partial}{\partial \mathbf{v}} \cdot (f \mathbf{v}) &= \\
= \frac{\partial f}{\partial t} + \frac{\partial}{\partial \mathbf{x}} \left( f \frac{\partial H}{\partial \mathbf{v}} \right) - \frac{\partial}{\partial \mathbf{v}} \left( f \frac{\partial H}{\partial \mathbf{x}} \right) &= \\
= \frac{\partial f}{\partial t} + \frac{\partial f}{\partial \mathbf{x}} \frac{\partial H}{\partial \mathbf{v}} - \frac{\partial f}{\partial \mathbf{v}} \frac{\partial H}{\partial \mathbf{x}} + f \frac{\partial^2 H}{\partial \mathbf{x} \partial \mathbf{v}} - f \frac{\partial^2 H}{\partial \mathbf{v} \partial \mathbf{x}} &= \\
= \frac{\partial f}{\partial t} + \dot{\mathbf{x}} \frac{\partial f}{\partial \mathbf{x}} + \dot{\mathbf{v}} \frac{\partial f}{\partial \mathbf{v}} &= 0
\end{aligned} \tag{4.3}$$

which represents the so called *collisionless Boltzmann equation* which is often known with the name of *Vlasov equation*. The equation (4.3) can be written in many different forms but, surely, the most compact form is that obtained using the extended (to 6 dimensions) concept of the convective Lagrangian derivative which is

$$\frac{\mathrm{d}}{\mathrm{d}t} \equiv \frac{\partial}{\partial t} + \dot{\mathbf{w}} \cdot \frac{\partial}{\partial \mathbf{w}} = \frac{\partial}{\partial t} + \dot{\mathbf{w}} \cdot \nabla_{\mathbf{w}} \,. \tag{4.4}$$

Using equation (4.4), the Vlasov equation (4.3) can be rewritten in the following elegant form

$$\frac{\mathrm{d}f}{\mathrm{d}t} = 0 \ . \tag{4.5}$$

The above cited master equation is a generalization of (4.5) introducing the quantity $\Gamma[f]$, instead of zero, in its second member. Obtaining an explicit expression for the DF of a generic stellar system is very complicated. Nevertheless, the following result obtained firstly by Jeans in 1915 (see Jeans [55]), can significantly reduce the complexity of the operation:

> **Jeans Theorem** *Any steady-state solution $f(\mathbf{x}, \mathbf{v})$ of the equation 4.5 depends on the phase-space coordinates only thorough integrals of motion in the given potential and vice versa.*

We have already introduced the concept of integral of motion in section 1.2.2. Explicitly we can write the equation (1.13) as

$$\frac{\mathrm{d}I}{\mathrm{d}t} = \frac{\partial I}{\partial \mathbf{x}} \cdot \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} + \frac{\partial I}{\partial \mathbf{v}} \cdot \frac{\mathrm{d}\mathbf{v}}{\mathrm{d}t} = \tag{4.6}$$

$$= \mathbf{v} \cdot \frac{\partial I}{\partial \mathbf{x}} - \frac{\partial \phi}{\partial \mathbf{x}} \cdot \frac{\partial I}{\partial \mathbf{v}} \tag{4.7}$$

where we have considered the generic Hamiltonian $H = K(v,t) + \phi(\mathbf{x},t)$ where $\phi(\mathbf{x},t)$ is the background gravitational potential (relative to the "field stars " ) and $K(v,t)$ the kinetic energy of the test particle. Any steady state solution of (4.5) is such that

$$\dot{\mathbf{x}} \cdot \frac{\partial f}{\partial \mathbf{x}} + \dot{\mathbf{v}} \cdot \frac{\partial f}{\partial \mathbf{v}} = 0 \Rightarrow \mathbf{v} \cdot \frac{\partial f}{\partial \mathbf{x}} - \frac{\partial \phi}{\partial \mathbf{x}} \cdot \frac{\partial I}{\partial \mathbf{v}} = 0 \tag{4.8}$$

which is identical to equation (4.7). Therefore we have proven that the function $f(\mathbf{x}, \mathbf{v})$ is an integral of motion. If, then, $f$ is a function of $n$ integrals $I_1, I_2, ..., I_n$ we have

$$\frac{\mathrm{d}f}{\mathrm{d}t}[I_1, I_2, ..., I_n] = \frac{\partial f}{\partial I_1}\frac{\mathrm{d}I_1}{\mathrm{d}t} + \frac{\partial f}{\partial I_2}\frac{\mathrm{d}I_2}{\mathrm{d}t} + ... + \frac{\partial f}{\partial I_n}\frac{\mathrm{d}I_n}{\mathrm{d}t} = 0 \tag{4.9}$$

where we have used equation the definition given in (1.13) in the last passage. Equation (4.9) proves the vice versa part of the Jeans theorem.

### 4.1.1 Ergodic distribution functions

It is possible to show also that

> **Strong Jeans Theorem** *The DF of a steady-state stellar system in which almost all orbits are regular with non-resonant frequencies may be presumed to be a function only of three independent isolating integrals, which may be taken to be the actions.*

The strong Jeans theorem, in broad lines, let us focus our attention, in this work, to study the properties of a restrict field of systems characterized by a DF which is function of the Hamiltonian $H = \frac{1}{2}v^2 + \phi(\mathbf{x}, t)$ only, which, in a steady-state potential $\phi(\mathbf{x})$, is an isolating integral of motion. These kinds of DF are said to be *ergotic*. As shown in [18] the mean velocity of systems described by an ergotic DF vanishes everywhere

$$\overline{\mathbf{v}}(r) = \frac{1}{\rho(r)} 4\pi \int \mathbf{v} v^2 f\left(\frac{1}{2}v^2 + \phi\right) = 0 \tag{4.10}$$

where

$$\rho(\mathbf{x}) \equiv \int \mathrm{d}^3\mathbf{v} f\left(\frac{1}{2}v^2 + \phi\right) \tag{4.11}$$

and $\mathbf{v} f\left(\frac{1}{2}v^2 + \phi\right)$ is an odd function of $\mathbf{v}$. The spread around $\overline{\mathbf{v}}(\mathbf{x}) = 0$ is characterized by the *velocity dispersion tensor* defined as

$$\sigma_{ij}^2(\mathbf{x}) \equiv \frac{1}{\rho(\mathbf{x})} \int \mathrm{d}^3\mathbf{v} (v_i - \overline{v_i})(v_j - \overline{v_j}) f\left(\frac{1}{2}v^2 + \phi\right) = \overline{v_i v_j} - \overline{v_i v_j} \tag{4.12}$$

which, in force of equation 4.10 reduces to

$$\sigma_{ij}^2(\mathbf{x}) = \frac{1}{\rho(\mathbf{x})}\int \mathrm{d}^3\mathbf{v} v_i v_j f\left(\frac{1}{2}v^2+\phi\right) = \overline{v_i v_j}\ . \tag{4.13}$$

Since $\sigma_{ij}(\mathbf{x})$ is symmetric, it is always possible to choose a base of vectors in which $\sigma_{ij}(\mathbf{x})$ is diagonal, that is $\sigma_{ij}^2(\mathbf{x}) = \sigma^2(\mathbf{x})\,\delta_{ij}$ therefore simplifying equation (4.13) we get

$$\sigma_{ii}^2(\mathbf{x}) = \frac{1}{\rho(\mathbf{x})}\int \mathrm{d}^3\mathbf{v} v_i^2 f\left(\frac{1}{2}v^2+\phi\right) = \overline{v_i^2}\ . \tag{4.14}$$

From equation (4.14) it is evident that $\sigma_{xx}^2(\mathbf{x}) = \sigma_{yy}^2(\mathbf{x}) = \sigma_{zz}^2(\mathbf{x}) = \sigma^2(\mathbf{x})$ therefore it is possible to write

$$\sigma^2(r) = \frac{4\pi}{3\rho(r)}\int_0^\infty \mathrm{d}v v^4 f\left(\frac{1}{2}v^2+\phi\right)\ . \tag{4.15}$$

Therefore we conclude by saying that a system described by an ergodic DF has an isotropic velocity dispersion tensor with value given by equation (4.15). Because some elliptical galaxies, most globular clusters and the inner regions of most galaxies can be treated with good approximation as spherical systems, it is convenient to restrict our study to the DF of such kind of systems. Following the notation given in [18] we introduce the relative potential and the relative energy defined as

$$\psi \equiv -\phi + \phi_0 \qquad \mathcal{E} \equiv \psi - \frac{1}{2}v^2 \tag{4.16}$$

where $\phi_0 = 0$ if the system extends to infinity. The relative potential $\psi$ must satisfy the Poisson's equation

$$\nabla^2\psi = -4\pi G\rho \tag{4.17}$$

which in the case of spherically symmetric systems can be rewritten in spherical coordinates as

$$\frac{1}{r^2}\frac{\mathrm{d}}{\mathrm{d}r}\left(r^2\frac{\mathrm{d}\psi}{\mathrm{d}r}\right) = -4\pi G\rho(r) \tag{4.18}$$

where $\rho(r)$ represents the mass density distribution of the system. In spherical symmetry we can write the equation (4.11) as

$$\rho(r) = 4\pi \int_0^\infty \mathrm{d}v v^2 f\left(\psi - \frac{1}{2}v^2\right) . \tag{4.19}$$

It is useful to change the integration variable from $v$ to $\mathcal{E}$ considering that

$$v^2 = 2(\psi - \mathcal{E}) \Rightarrow \mathrm{d}v = -\frac{\mathcal{E}\mathrm{d}\mathcal{E}}{\sqrt{2(\psi - \mathcal{E})}} \tag{4.20}$$

which substituted in equation (4.19) give us

$$\rho(r) = -4\pi \int_\psi^0 2(\psi - \mathcal{E}) f(\mathcal{E}) \frac{\mathcal{E}\mathrm{d}\mathcal{E}}{\sqrt{2(\psi - \mathcal{E})}} = 4\pi \int_0^\psi \mathrm{d}\mathcal{E} f(\mathcal{E}) \sqrt{2(\psi - \mathcal{E})} \tag{4.21}$$

where we have chosen $\phi_0$ such that $f = 0$ for $\mathcal{E} \leq 0$ to obtain the new extremes of integration. In any spherical system it can be shown that $\psi$ is a monotonic function of $r$ therefore we can transform $\rho(r)$ in $\rho(\psi)$ and write

$$\frac{\rho(\psi)}{\sqrt{8}\pi} = 2\int_0^\psi \mathrm{d}\mathcal{E} f(\mathcal{E}) \sqrt{\psi - \mathcal{E}} . \tag{4.22}$$

Differentiating both sides with respect to $\psi$ we get

$$\frac{1}{\pi\sqrt{8}} \frac{\mathrm{d}\rho}{\mathrm{d}\psi} = 2\frac{\mathrm{d}}{\mathrm{d}\psi} \int_0^\psi \mathrm{d}\mathcal{E} f(\mathcal{E}) \sqrt{\psi - \mathcal{E}} = \int_0^\psi \mathrm{d}\mathcal{E} \frac{f(\mathcal{E})}{\sqrt{\psi - \mathcal{E}}} \tag{4.23}$$

where we have used the *Leibniz integration rule* [43] and the fact that $f(\mathcal{E} = 0) = 0$. We can extract the DF from the integral noting that equation (4.23) is an *Abel integral equation*. In fact, having a generic function

$$g(x) = \int_0^x \frac{\mathrm{d}t s(t)}{(t - x)^\alpha} \quad (0 < \alpha < 1) \tag{4.24}$$

it is possible to show [31] that

$$s(t) = -\frac{\sin(\pi\alpha)}{\pi} \frac{\mathrm{d}}{\mathrm{d}t} \int_0^t \frac{\mathrm{d}x g(x)}{(x - t)^{(1-\alpha)}} . \tag{4.25}$$

In our case $\alpha = \frac{1}{2}$, $t = \mathcal{E}$, $x = \psi$ and $g(x) = \frac{1}{\pi\sqrt{8}} \frac{\mathrm{d}\rho}{\mathrm{d}\psi}$ therefore equation 4.23 may be transformed in

$$f(\mathcal{E}) = \frac{1}{\pi^2\sqrt{8}}\frac{\mathrm{d}}{\mathrm{d}\mathcal{E}}\int_0^{\mathcal{E}}\frac{\mathrm{d}\psi}{\sqrt{\mathcal{E}-\psi}}\frac{\mathrm{d}\rho}{\mathrm{d}\psi}\,. \tag{4.26}$$

Eddington [40] was the first to obtain this result in fact equation (4.26) is often referred as *Eddington's formula*. This is a very important result because using equation (4.26) we can always obtain a DF for a system with explicitly known mass density profile $\rho(r)$ provided that the integral

$$\int_0^{\mathcal{E}}\frac{\mathrm{d}\psi}{\sqrt{\mathcal{E}-\psi}}\frac{\mathrm{d}\rho}{\mathrm{d}\psi} \tag{4.27}$$

is an increasing function of $\mathcal{E}$in order to guarantee $f(\mathcal{E}) > 0\ \forall\mathcal{E} > 0$. If not, the DF for the considered mass density profile cannot be ergodic.

### 4.1.2 The Plummer distribution function

A very simple example of the application of the formula (4.26) to a case of astrophysical practical interest is the procedure to obtain the DF of the Plummer model [82]. This model is characterized by a potential given by

$$\phi_p(r) = -\frac{GM}{\sqrt{r^2+b^2}} = -\frac{GM}{b}\left(1+\frac{r^2}{b^2}\right)^{-\frac{1}{2}} \tag{4.28}$$

where $M$ is the total mass of the system and $b$ is the so called Plummer core radius. To obtain the corresponding density profile we have to solve the Poisson's equation in spherical symmetry (4.18) obtaining

$$\rho_p(r) = \frac{3M}{4\pi b^3}\left(1+\frac{r^2}{b^2}\right)^{-\frac{5}{2}}\,. \tag{4.29}$$

From equation (4.28) we can get

$$\left(1+\frac{r^2}{b^2}\right)^{-\frac{5}{2}} = -\left[\frac{b}{GM}\phi_p(r)\right]^5 \tag{4.30}$$

which substituted into equation (4.29) gives us

$$\rho_p\left(\psi_p\right) = \frac{3}{4\pi}\frac{b^2}{G^5M^4}\psi_p^5 \ . \tag{4.31}$$

The derivative of $\rho_p\left(\psi_p\right)$ with respect to $\psi_p$ writes

$$\frac{\mathrm{d}\rho_p}{\mathrm{d}\psi_p} = \frac{15}{4\pi}\frac{b^2}{G^5M^4}\psi_p^4 \tag{4.32}$$

which, substituted into equation (4.26) gives us the following expression for the Plummer DF

$$f_p\left(\mathcal{E}\right) = \frac{15}{8\pi^3\sqrt{2}}\frac{b^2}{G^5M^4}\frac{\mathrm{d}}{\mathrm{d}\mathcal{E}}\int_0^{\mathcal{E}}\frac{\mathrm{d}\psi\psi^4}{\sqrt{\mathcal{E}-\psi}} \ . \tag{4.33}$$

One method to solve the integral which appears in equation (4.33) is to integrate firstly four times by parts and then solve a known standard integral of the form $\int f\left(x\right)^{\alpha}\mathrm{d}x$. The result is

$$\int_0^{\mathcal{E}}\mathrm{d}\psi\frac{\psi^4}{\sqrt{\mathcal{E}-\psi}} = \frac{256}{315}\mathcal{E}^4\sqrt{\mathcal{E}} \tag{4.34}$$

which can be derived with respect to $\mathcal{E}$ letting us obtain the following complete expression of the Plummer DF

$$f_p\left(\mathcal{E}\right) = \frac{24\sqrt{2}}{t\pi^3}\frac{b^2}{G^5M^4}\mathcal{E}^{\frac{7}{2}} \ . \tag{4.35}$$

It is possible to follow an analogous procedure to obtain DF for other models with a known shape of the mass density profile and/or the potential. In particular, the so called Dehnen models [38] are of practical astrophysical interest because their associated gravitational potential is analytical and, for some of them, it is possible to get to an explicit expression of the DF . They are very useful and used to describe and model a wide sample of elliptical galaxies (for the detail see Dehnen [38]).

### 4.1.3 The King distribution function

A very important class of models which are very useful in astrophysics especially because they approximate quite well most of the density profiles of most globular clusters are the so called *lowered isothermal models* to which the King's profiles belong [56]. These models are born to resemble the isothermal distribution at small distances from the centre of mass of the system while, the density profile falls to zero more rapidly at large distances ensuring the good property of having a finite total mass for the entire system. This class of profiles is obtained starting from a modified version of the DF of a isothermal sphere requiring that $f(\mathcal{E}) = 0$ for $\mathcal{E} \leq \mathcal{E}_0$ where $\mathcal{E}_0$ is also known as the *critical relative energy*. The explicit expression for the King DF , for values of $\mathcal{E}$such that $\mathcal{E} > \mathcal{E}_0$, is

$$f_k(\mathcal{E}) = \rho_1 \left(2\pi\sigma^2\right)^{-\frac{3}{2}} \left(e^{\frac{\mathcal{E}}{\sigma^2}} - 1\right) . \tag{4.36}$$

In particular, it is always possible to choose the constant $\phi_0$, which appear in the definition of the relative potential, such that $\mathcal{E}_0 = 0$. To get to the density profile it is sufficient to substitute equation (4.36) in the expression given in (4.21) and use the definition of relative energy, obtaining

$$\rho_k(\psi_k) = 4\pi \int_0^{\psi_k} \mathrm{d}\mathcal{E}\, \rho_1 \left(2\pi\sigma^2\right)^{-\frac{3}{2}} \left(e^{\frac{\mathcal{E}}{\sigma^2}} - 1\right) \sqrt{2(\psi - \mathcal{E})} \tag{4.37}$$

which, integrated, gives us the formula

$$\rho_k(\psi_k) = \rho_1 \left[ e^{\frac{\psi_k}{\sigma^2}} \operatorname{erf}\left(\frac{\sqrt{\psi_k}}{\sigma}\right) - \sqrt{\frac{4\psi_k}{\pi\sigma^2}} \left(1 + \frac{2\psi_k}{3\sigma^2}\right) \right] . \tag{4.38}$$

The Poisson's equation for such models it is usually written introducing new dimensionless variables $\widetilde{\rho}$ and $\widetilde{r}$, in the place of $\rho$ and $r$, which are defined in terms of the central density $\rho_0$ and the so called *King radius* $r_0$

$$\widetilde{\rho} \equiv \frac{\rho_k}{\rho_0} \qquad \widetilde{r} \equiv \frac{r}{r_0} \qquad r_0 \equiv \sqrt{\frac{9\sigma^2}{4\pi G \rho_0}} \,. \tag{4.39}$$

These variables were introduced, for the first time, to describe the singular isothermal sphere whose density profile is singular at $r = 0$ while the trend of $\widetilde{\rho}\,(\widetilde{r})$ is well behaved at the origin. Using the new variables, the Poisson's equation writes

$$\frac{1}{\widetilde{r}^2}\frac{\mathrm{d}}{\mathrm{d}\widetilde{r}}\left(\widetilde{r}^2\frac{\mathrm{d}\psi}{\mathrm{d}\widetilde{r}}\right) = -9\sigma^2\widetilde{\rho}\,. \tag{4.40}$$

For the King model we introduce now another parameter, $W$, fundamental in every King's model, defined as

$$W = \frac{\psi_k}{\sigma^2} \tag{4.41}$$

often called *dimensionless potential*. Substituting the expression for $W$ in equation 4.40 and carrying out the derivatives we get

$$\frac{\mathrm{d}^2 W}{\mathrm{d}\widetilde{r}^2} + \frac{2}{\widetilde{r}}\frac{\mathrm{d}W}{\mathrm{d}\widetilde{r}} + 9\widetilde{\rho}\,(W, W_0) = 0 \tag{4.42}$$

where

$$\widetilde{\rho}\,(W, W_0) = \frac{e^W \mathrm{erf}\left(\sqrt{W}\right) - \sqrt{\frac{4}{\pi}W}\left(1 + \frac{2}{3}W\right)}{e^{W_0}\mathrm{erf}\left(\sqrt{W_0}\right) - \sqrt{\frac{4}{\pi}W_0}\left(1 + \frac{2}{3}W_0\right)}\,. \tag{4.43}$$

The solution of equation (4.42) can be obtained numerically provided that appropriate initial conditions for $W$ and $\dot{W}$ are chosen. Specifically, the value of $W\,(\widetilde{r} = 0) = W_0$ represents the depth of the central potential well, and it is needed to require that its first derivative with respect to $\widetilde{r}$, calculated at null distance, is null $\left(\dot{W}_0 = 0\right)$. The latter condition corresponds to have a null force at the centre of the system which, indeed, is in perfect agreement with its spherical symmetry and, moreover, corresponds to have a finite (null) mass ar $\widetilde{r} = 0$. The solution for $W\,(\widetilde{r})$ is such that $W\,(\widetilde{r}) \Leftrightarrow \widetilde{r} \geq r_t$ where $r_t$ is called *tidal radius* of the King model which represents also the distance at which the density vanishes.

In general, the bigger the value of the central potential well ($W_0$) the greater the tidal radius will be. Unfortunately, it is straightforward that it is not always possible to get an explicit expression for the DF of a system which has a generic, spherically symmetric, $\rho(r)$. The most general way is to use numerical techniques to solve directly equation (4.26) given a certain formula (or numerical evaluation) of the density profile as a function of the distance (or, equivalently, directly of the gravitational potential). The procedure to obtain the DF for a spherical system is very important because it constitutes one of the main steps to get a computer model of a $N$-body system. In broad lines, what we need to do is to obtain positions, velocities and, of course, masses for all the bodies belonging to the astrophysical system.

## 4.2 Generating initial conditions

### 4.2.1 Positions

In order to perform this step, it is fundamental to introduce the cumulative mass distribution defined as

$$M(r) = \int_0^r 4\pi r^2 \rho(r)\,\mathrm{d}r = -\frac{r^2}{G}\frac{\mathrm{d}\phi(r)}{\mathrm{d}r} \tag{4.44}$$

which expresses the total mass enclosed in a sphere of radius $r$. This represents a cumulative distribution function of the probability density function $\mathrm{d}p(r) = 4\pi r^2 \rho(r)\,\mathrm{d}r$ which represents, indeed, the probability to find a star in a volume extended between $r$ and $r + \mathrm{d}r$. Therefore, what we can do is to invert the mass distribution function in order to obtain the position $r(M_{rand}))$ starting from a randomly generated number, $M_{rand}$, between $M_{min}$ [2] and $M_{max} = M$.

[2] In general, $M_{min} \neq 0$. In fact, each star of the system has its own mass $m_{star}$, therefore, $M_{min} \geq m_{star}$. Moreover, sometimes, there can be numerical difficulties to integrate Poisson's equation exactly form/to $r = 0$.

## 4.3 Velocities

Having determined the position of the $i$-th particle ($r_i$), to assign it a proper velocity, we need to know the DF of the system. First of all it is necessary that

$$v_i \in [0; v_e(r_i)] \tag{4.45}$$

where $v_e(r_i)$ represents the escape velocity at distance $r_i$ which can be written in terms of the relative potential $\psi(r)$

$$v_e(r_i) = \sqrt{2\psi(r_i)}\ . \tag{4.46}$$

The probability distribution for the velocities is strictly linked with the DF . In fact, the probability to have an absolute value for the velocity, of the $i$-th particle between $v_i$ and $v_i + \mathrm{d}v_i$ at position $r_i$ is given by the probability density function

$$\mathrm{d}p(v_i; r_i) = 4\pi v_i^2 f\left(\psi(r_i) - \frac{1}{2}v_i^2\right)\mathrm{d}v_i\ . \tag{4.47}$$

In principle, one can follows the same procedure discussed to sample the positions of the particles; in fact, the DF $f\left(\psi(r_i) - \frac{1}{2}v_i^2\right)$ is completely equivalent to the role of $\rho(r)$ therefore to sample velocities in the right way, it is possible to invert (numerically) the relation

$$\nu(v) = \int_0^v 4\pi v^2 f(\mathcal{E})\,\mathrm{d}v \tag{4.48}$$

extracting random numbers for $\nu(r)$ from $\nu_{min}$ and $\nu_{max}$. Nevertheless this procedure is quite difficult because while the function $M(r)$ can be easily calculated explicitly or, at least, numerically tabulated while solving the Poisson's equation through the evaluation of the derivative of $\psi$, the function $\nu(r)$ is quite laborious to obtain. In this case it is easier to proceed using the so called method of the *Acceptance and Re-*

Therefore, if the minimum distance reached by the integration is $r_{min} = \epsilon$ the minimum mass must be chosen such that $M_{min} = M(r_{min}) \geq m_{star}$.

*jection* directly on the expression of the DF which has to be calculated, in any case, using the relation (4.26). What we can do is following this schematic procedure

1. given a particle at position $r_i$, the limits on the allowed energies for this body must be determined. The interval is such that $\mathcal{E}_i \in [0, \psi(r_i)]$ where $\mathcal{E}_i = 0$ corresponds to $v_i = v_e(r_i)$ and $\mathcal{E}_i = \psi(r_i)$ is equivalent to $v_i = 0$;

2. determine the minimum value $(f_{min})$ and the maximum value $(f_{max})$ of the DF in the interval of all the possible energies obtained in the previous schematic step;

3. continue to extract a random number for the velocity $v_i \in [0, v_e(r_i)]$ until another random number $f_1 \in [0, f_{max}]$ becomes smaller than $f_0 = f\left(\psi(r_i) - \frac{1}{2}v_i^2\right)$. This is the main part of the acceptance/rejection technique;

4. Choose the extracted $v_i$ as velocity for the particle in position $r_i$.

This schematic representation of the problem works theoretically fine and it helps us to understand the overall procedure to implement a code which generates stable initial conditions for a generic $N$-body system in spherical symmetry.

## 4.4 Numerical implementation

### 4.4.1 Initial conditions for $\psi$ and $\psi^{'}$

We implemented a code which samples a generic spherical stellar system starting from the expression of its mass density distribution. Although some other similar implementations already exist (see for example [92] and also [60]) we preferred to implement our own version which is very easy to use, it can sample the $N$-body computer model starting from any spherical density profile and it can also create a stable stellar system containing a super massive black hole (SMBH). The first thing to do is to solve the Poisson's equation in order to obtain $\psi(r)$, $\psi^{'}(r)$ and, consequently, $M(r)$ from the expression (4.44). In general, one does not know much about initial conditions at $r = 0$ which are $\psi(0) = A$ and $\psi^{'}(0) = B$. Nevertheless, it is true that, if we have a stellar system with characteristic dimension $R$ and we see it from a distance $r \gg R$, we can consider it a point of mass, therefore we can write

$$\psi(r \gg R) \simeq \frac{GM}{r} \qquad \psi^{'}(r \gg R) \simeq -\frac{GM}{r^2} \quad . \tag{4.49}$$

where $M$ is the total mass of the system. Therefore, it is convenient to start the numerical integration form $r \gg R$ back to $r \simeq 0$ in order to know the initial conditions for the for the gravitational potential and its first derivative. It is also clear that the total mass of the system (to be precise, the mass at $r \gg R$) must be previously determined solving numerically the integral in equation 4.44. The exception is represented by the King model which has $\psi(r \geq r_t) = 0$ therefore it is needed an integration form $r = 0$ to $r = r_t$ choosing $\psi(0) = W_0\sigma^2$ and $\psi^{'}(0) = 0$ that is the initial values of $W_0$ and $\sigma^2$ (or equivalently the King's radius $r_0$, see equation 4.39) must be chosen. It is also straightforward that a very precise integrator is needed because we need to integrate Poisson's equation over something like dozens of orders of magnitude in

distance (let us say from $r \simeq 10^{-10}R$ to $r \simeq 10^{10}R$) maintaining very good accuracy and, above all, in a reasonable wall clock (human) time. To integrate it we choose the Bulirsch-Stoer method (see for example [84]).

### Notes about the Bulirsch-Stoer integrator

This method constitutes one of the best algorithms to obtain very high accuracy with minimal computational effort provided that the functions envolved in the problem are neither singular or complex to evaluate. The main idea at the base of the BS method is the so called *Richardson extrapolation* which thinks the final solution of a numerical problem as itself being a function of the time step ($h$ in our case) used to get it. The key idea is to choose a so called macro time step, $\Delta r$ and obtain different solutions of $\psi(r_0 + \Delta r)$ using several values for the time step such that

$$h_n = \frac{\Delta r}{n} \tag{4.50}$$

with, for example, $n = 2, 4, 6, 8, 10, 12, 14 \ldots$. In this way we have

$$\psi(r_0 + \Delta r) = \lim_{h \to 0} g(h) \tag{4.51}$$

where $g(h)$ is a function obtained interpolating the $n$ different solution attempts for $\psi(r_0 + \Delta r)$. In particular, to interpolate points in our code we used a rational function extrapolation. To advance the solution with steps $h_n$ we used the so called *Modified Midpoint Method* (MMM) which is not very accurate (second order) but coupled with the Richardson technique is proven to be very powerful because of the result obtained by Grass [46] which showed that the error of the MMM can be expressed as a power series of the time step which contains only its even powers. This means that each following Richardson attempt is more precise with respect to the previous one by 2 orders. To adapt the MMM to the Pois-

son's equation we need to write it as the combination of two differential equations of the first order:

$$\begin{cases} \psi^{'} = \zeta \\ \zeta^{'} = -4\pi G\rho\left(r,\psi,\zeta\right) - \frac{2}{r}\zeta = K\left(r,\psi,\zeta\right) \\ \psi_0 = A \\ \zeta_0 = B. \end{cases} \tag{4.52}$$

The following steps represent schematically the MMM applied to the solution of 4.52 to advance it from $r_0$ to $r_0 + \Delta r$ using a time step $h = \frac{\Delta r}{n}$; we will use the notation $r_i = r_0 + ih$:

$$\begin{aligned} \psi\left(r_0\right) &= A \\ \zeta\left(r_0\right) &= B \\ t_0 &= K\left(r_0, \psi\left(r_0\right), \zeta\left(r_0\right)\right) \\ \zeta\left(r_1\right) &= \zeta\left(r_0\right) + ht \\ \psi\left(r_1\right) &= \psi\left(r_0\right) + h\zeta\left(r_0\right) \end{aligned} \tag{4.53}$$

$$\begin{cases} t_i = K\left(\left(r_i, \psi\left(r_i\right), \zeta\left(r_i\right)\right)\right. \\ \zeta\left(r_{i+1}\right) = \zeta\left(r_{i-1}\right) + 2ht_i, \quad i = 1, 2, ...n-1 \\ \psi\left(r_{i+1}\right) = \psi\left(r_{i-1}\right) + h\zeta\left(r_i\right) \end{cases} \tag{4.54}$$

$$\begin{aligned} \psi\left(r_0 + \Delta r\right) &= \frac{1}{2}\left[\psi\left(r_n\right) + \left(\psi\left(r_{n-1}\right) + h\zeta\left(r_n\right)\right)\right] \\ t_n &= K\left(r_n, \psi\left(r_n\right), \zeta\left(r_n\right)\right) \\ \psi^{'}\left(r_0 + \Delta r\right) &= \zeta\left(r_0 + \Delta r\right) = \frac{1}{2}\left[\zeta\left(r_n\right) + \left(\zeta\left(r_{n-1}\right) + ht_n\right)\right] \ . \end{aligned} \tag{4.55}$$

Simultaneously, $M\left(r\right) = -\frac{r^2}{G}\psi^{'}\left(r\right)$ can also be obtained. For a more detailed description about the BS method or the MMM it is possible to see [84]. Once the Poisson's equation has been solved, the analytic contribution of a central black hole (BH) with mass $M_{BH}$ can be included, adding its contributions to $\psi\left(r\right), \psi^{'}\left(r\right)$ and $M\left(r\right)$.

### 4.4.2 The evaluation of $\frac{\mathrm{d}\rho}{\mathrm{d}\psi}$

The next step is the evaluation of the quantity $\frac{\mathrm{d}\rho}{\mathrm{d}\psi}$ which is needed to obtain the numerical evaluation of the DF from equation (4.26). We need a method to estimate the first derivative from a certain number of tabulated points of $\psi(r)$ and $\rho(r)$. We verified that the simple approach

$$\frac{\mathrm{d}\rho}{\mathrm{d}\psi}(\psi_i) = \frac{\rho(\psi_i + \Delta\psi) - \rho(\psi_i)}{\Delta\psi} \tag{4.56}$$

was not enough accurate to guarantee sufficient precision for the following steps to execute. The same can be said for the slightly different approach

$$\frac{\mathrm{d}\rho}{\mathrm{d}\psi}(\psi_i) = \frac{\rho(\psi_i - \Delta\psi) - \rho(\psi_i + \Delta\psi)}{2\Delta\psi}\,. \tag{4.57}$$

In fact, a further complication which arises, in our case, is that the tabulated values of $\psi_i$ are not, obviously, monospaced, that is $\Delta\psi \neq \mathrm{constant}$, therefore we need to get to a more general approach to evaluate the first derivative needed. A simple, but valid, idea is to start with the Taylor expansions of the function $\rho(\psi)$ in 3 points

$$\begin{cases} \rho(\psi_0 + h_1) = \rho(\psi_0) + \rho'(\psi_0)\, h_1 + \frac{1}{2}\rho''(\psi_0)\, h_1^2 + \frac{1}{6}\rho'''(\psi_0)\, h_1^3 \\ \rho(\psi_0 - h_2) = \rho(\psi_0) - \rho'(\psi_0)\, h_2 + \frac{1}{2}\rho''(\psi_0)\, h_2^2 - \frac{1}{6}\rho'''(\psi_0)\, h_2^3 \\ \rho(\psi_0 + h_3) = \rho(\psi_0) + \rho'(\psi_0)\, h_3 + \frac{1}{2}\rho''(\psi_0)\, h_3^2 + \frac{1}{6}\rho'''(\psi_0)\, h_3^3. \end{cases} \tag{4.58}$$

In this way we have a system composed by 3 equations in 3 unknowns $\rho'(\psi_0)$, $\rho''(\psi_0)$ and $\rho'''(\psi_0)$ therefore we can obtain an explicit expression of $\rho'_+(\psi_0)$. The latter expression can be also averaged, for example, with another evaluation of the first derivative, $\rho'_-(\psi_0)$, using the expansion of $\rho(\psi_0 - h_4)$ instead of $\rho(\psi_0 + h_3)$ in system (4.58).

### 4.4.3 The numerical evaluation of $f(\mathcal{E})$

The next step is to create a grid of energies (we verified that a logarithmic grid gives better results) to evaluate the DF using equation 4.26. The number of points in the grid must be determined for each specific case but we found that, in general, a value of 5000 points represents a good compromise between speed of the integration and accuracy of the DF . Nevertheless, the integrand which appears in equation 4.26 is numerically problematic in the extreme $\psi = \mathcal{E}$ (even if, from an analytic point of view, the integrand can be convergent due to the trend of $\frac{\mathrm{d}\rho}{\mathrm{d}\psi}$). To avoid numerical problems, for each value of energy $\widetilde{\mathcal{E}}$ in the grid, we integrate, with a certain step $\mathrm{d}\widetilde{\mathcal{E}_1}$, from 0 up to $\widetilde{\mathcal{E}} - \mathrm{d}\widetilde{\mathcal{E}_1}$. Then, we reduce the integration step $\mathrm{d}\widetilde{\mathcal{E}_2} = \eta \mathrm{d}\widetilde{\mathcal{E}_1}$ (we chose $\eta \lesssim 10^{-3}$) and we begin another integration from $\widetilde{\mathcal{E}} - \mathrm{d}\widetilde{\mathcal{E}_1}$ to $\widetilde{\mathcal{E}} - \mathrm{d}\widetilde{\mathcal{E}_2}$. The result of the latter integration is added as a "corrective" term to the first evaluation of the integral. We iterate this procedure adding more corrective terms until the numerical value of the integral does not vary significantly any more adding further correction terms. To evaluate each integral we used the *Simpson method* refined with the Richardson extrapolation method (see section 4.4.1). To obtain the DF , a derivative of the just evaluated integral has to be calculated; this is performed using the same strategy shown before to evaluate $\frac{\mathrm{d}\rho}{\mathrm{d}\psi}$. Having these quantities, the procedure schematically described before can be followed and iterated in order to generate positions and velocities for all the stars in the $N$-body system.

## 4.5 Time Units

Any $N$-body computer model which derives from the application of a generic "sampling " code, is just a collection of numbers which are not directly linked with physical units such as solar masses, km/s, par-

secs and so on. To sample a generic computer model, astrophysicists, in general, for pure convenience, use to put $G = 1$ and the same is done in most $N$-body codes. To refer to a real astrophysical systems it is important to choose a scale distance $(R_s)$ and a scale mass $(M_s)$ and, from them, it is possible to obtain a time scale unit $(T_s)$ and a scale velocity $(V_s)$ in order to completely characterize the sampled stellar system. The physical quantities can be obtained using the relations $m_{phys} = M_s m_{num}$ and $r_{phys} = R_s r_{num}$ and it is straightforward to show that the time unit is

$$T_s = 14.9477133878319 \frac{R_s^{\frac{3}{2}}/\mathrm{pc}}{\sqrt{M_s/M_\odot}} \ \mathrm{Myr} \tag{4.59}$$

where the factor 14.92 comes from the value of $\frac{1}{\sqrt{G}}$ in units of parsecs, solar masses and mega years. The same can be said for the scale velocity which is

$$V_s \simeq 6.54589713446219 \times 10^{-2} \sqrt{\frac{M_s/M_\odot}{R_s/\mathrm{pc}}} \ \mathrm{km/s} \tag{4.60}$$

therefore the physical quantities can be obtained using the relations $t_{phys} = T_s t_{num}$ and $v_{phys} = V_s v_{num}$. Sometimes, it is convenient to use the so called $N$-body units (see for example [2]) which are characterized by having $G = M = R_V = 1$ where $R_V$ is the so called virial radius of the system which is such that

$$\frac{1}{R_V} = 2 \sum_{i \neq j}^{N} \frac{m_i m_j}{r_{ij}} \ . \tag{4.61}$$

It is possible to show that, in these units, the total energy of the system is $E = -\frac{1}{4}$. The use of $N$-body units is widespread but their usage does not constitute a rule. Moreover, they cannot deal with those systems having a positive total energy. A good general rule is to choose $M_s$ and $R_s$ in order to obtain a time unit $T_s$ which guarantee a "regular" distribution of time steps in $N$-body codes, like `HiGPUs` , that use block

time steps. "Regular" means that the distribution of time steps for a generic system should extend from about $2^{-15}$ to $2^{-3}$ resembling a bell shape (like a gaussian distribution), If the majority of time steps are around $2^{-3}$ means that the time unit should be increased in order to lower time steps. The contrary can be said for time steps which are, on average, too low.

## 4.6 Practical tests

This section shows $N$-body computer models obtained using our new implementation of the code which can sample a stellar system starting from a generic density profile in spherical symmetry. We use as tests cases a Plummer model (`P1`), a King model (`K1`), a Dehnen model (`D1`) and a custom model (`C1`) which also includes a central super massive black hole (SMBH). All the studied systems, except the system `C1`, have total, numerical, mass $M = 1$ and we also use $G = 1$. The Plummer core radius ($b$) and the King core radius ($r_0$) are such that $b = r_0 = 1$ while the King's central dimensionless potential is $W_0 = 7$. For the Dehnen profile, characterized by a mass density of the form

$$\rho_{\texttt{D1}}(r) = \frac{(3-\gamma)\,M}{4\pi}\frac{a}{r^{\gamma}\,(r+a)^{(4-\gamma)}}, \tag{4.62}$$

the parameter $\gamma$ has been chosen such that $\gamma = 1$ and, similarly, the Dehnen's scale length $a$ is also unitary. The system `C1` is characterized by a Dehnen ($\gamma = 0.2$) density profile truncated at a certain scale radius ($r_{cut}$) with an exponential term included in the function sech(x). This profile is such that

$$\rho_{\texttt{C1}}(r) = \frac{7M}{10\pi}\frac{a}{r^{0.2}\,(r+a)^{3.8}}\mathrm{sech}\left(\frac{r}{r_{cut}}\right). \tag{4.63}$$

The possibility to sample a $N$-body model starting from a generic spherical density profile constituted the main motivation to implement our

new code. Specifically, the density profile expressed in equation (4.63) is very important for astrophysical simulations for several reasons. Suppose, for example, that we want to study the dynamical evolution of the innermost region of a generic elliptical galaxy which, globally, is well described by a Dehnen, $\gamma = 2$ density profile. Suppose also that the total mass of the galaxy is $M_{gal} = 10^{11} M_{\odot}$ and that we would like to concentrate our simulation on the first 50 parsecs of this object. Within this distance, the galaxy contains, considering, for example, $a = 2\mathrm{kpc}$, $M_{gal}\,(r = 50\mathrm{pc}) \sim 5 \times 10^{7} M_{\odot}$ which, on average, means something like $10^{7} \div 10^{8}$ stars. Nevertheless it is not possible to sample the entire galaxy with a Dehnen model and then select only the particles in the sphere of radius $r = 50\mathrm{pc}$ for, at least, two reasons:

1. for numerical reasons, the maximum, reasonable, number of particles that can be dynamically evolved, at least using a direct summation $N$-body code, must be $N \lesssim 2\mathrm{M}$. This means that if we want to sample the above described galaxy using this number of bodies, each star has to have a mass of about $m_{star} \simeq \frac{10^{11}}{2\mathrm{x}10^{6}} M_{\odot} \simeq 5 \times 10^{4} M_{\odot}$ which is significantly not realistic. This results comes directly from the fact that we have tried to sample a system containing, in reality, $10^{11}$ stars using "only" $10^{6}$ particles. Moreover, the spatial resolution of the simulation, especially in the very dense central regions, is poor therefore this kind of under-sampling process should be avoided;

2. One strategy could be to perform on oversampling of the entire model using for example $N \sim 10^{10} \div 10^{11}$ stars and then truncate this model at $r \sim 50\mathrm{pc}$ operating a brute selection of particles. Nevertheless, in this way, it is straightforward to understand and verify that the resulting system is not stable (it will tend to expand very fast because we have eliminated an entire region of the original, stable, phase space).

The possible solution consists to modify the density profile such that at small radii ($r \lesssim 50\mathrm{pc}$) resembles the Dehnen's distribution while it falls rapidly to zero for $r \gtrsim 50\mathrm{pc}$ which is exactly the behaviour of the truncated density profile shown in equation 4.63. In this way we have a self consistent model truncated at $r_{cut} = 50\mathrm{pc}$ which let us concentrate our attention on a specific region of the galaxy which we want to study. In the last part of this chapter we will use the system `C1` to show the stability of the computer models obtained using our numerical implementation of the "sampling" problem. Now we focus our attention to study the properties obtained for the models `D1`, `K1` and `P1`.

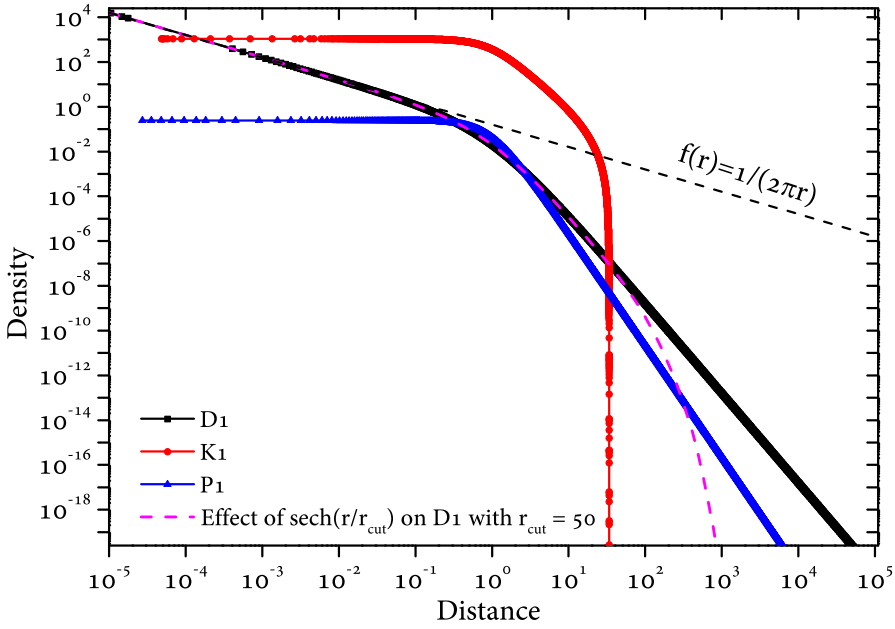


**Figure 4.1:** Density profiles obtained for the systems `D1`, `K1` and `P1`. The points are the numerical output of our code, continuous lines are the theoretical expressions. The pink dashed line is reported to show the effect of the function sech(x) on the density profile of the model D1. The black dashed line is shown to underline the divergence of the Dehnen density profile for $r \sim 0$ which goes as $1/r$.

Fig. 4.1 shows the density profiles of the tested systems. The systems present deep differences from one another; first of all, the `D1` model

(black points) shows a cusp for $r \sim 0$ which goes to infinity as the function $f(r)$ represented by a black dashed line ($\propto 1/r$) while the models K1 (red points) and P1 (blue points) are characterized by the presence of a flat core ($\rho_{\texttt{K1}} \sim$ const. for $r \lesssim r_0$ and $\rho_{\texttt{P1}} \sim$ const. for $r \lesssim b$). It is also evident what we have already pointed out about the King density profile. It is such that $\rho_{\texttt{K1}}(r) = 0$ for $r \geq r_t$ and, in this case ($W_0 = 7$ and $r_0 = 1$), $r_t = 33.708$ which is in agreement to what already shown in [56]. In Fig.4.1 we show also (pink dashed line) the analytic form of the model D1 truncated with the function $\mathrm{sech}\left(\frac{r}{r_{cut}}\right)$ with $r_{cut} = 50$. As expected, it resembles the system D1 for $r \lesssim r_{cut}$ while it falls rapidly (exponentially) to zero for $r \gtrsim r_{cut}$.

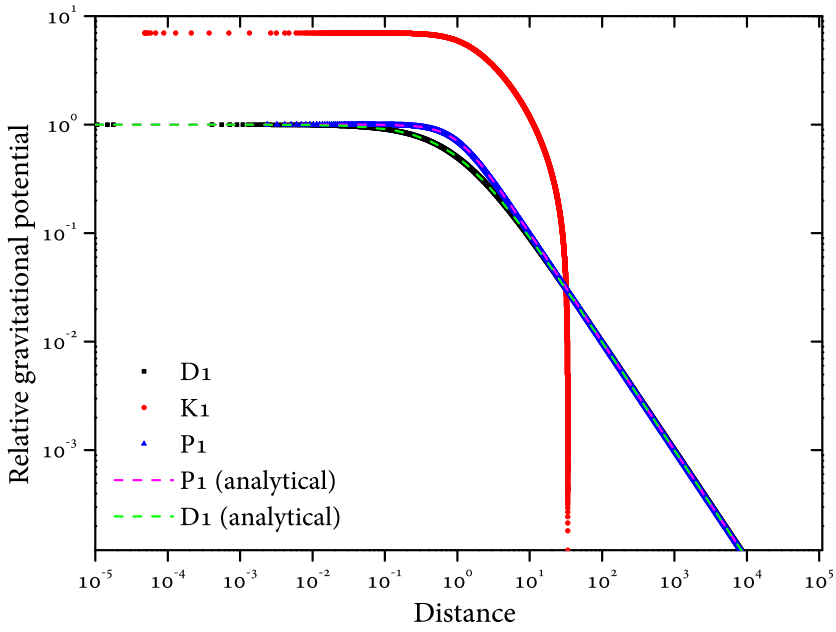


**Figure 4.2:** Gravitational potential for the tested models resulting from the numerical integration of the Poisson's equation. The exact expressions (dashed lines) are shown for the models D1 adn P1 but not for the King model for which the expression for the gravitational potential cannot be written explicitly.

Fig. 4.2 shows the gravitational potential (points) obtained from the numerical solution of the Poisson's equation for systems `D1`, `P1` and `K1`. The dashed lines represent the analytical, explicit forms of the potentials (except for the King model for which the solution is numerical only). As we can see, the analytical trends are not distinguishable, by eye, from the points obtained in the integration which extends up to a distance of $\sim 10^{11}$ although we decided to cut the Fig. 4.2 at $r \simeq 10^5$ in order to show a more clear representation of the results.

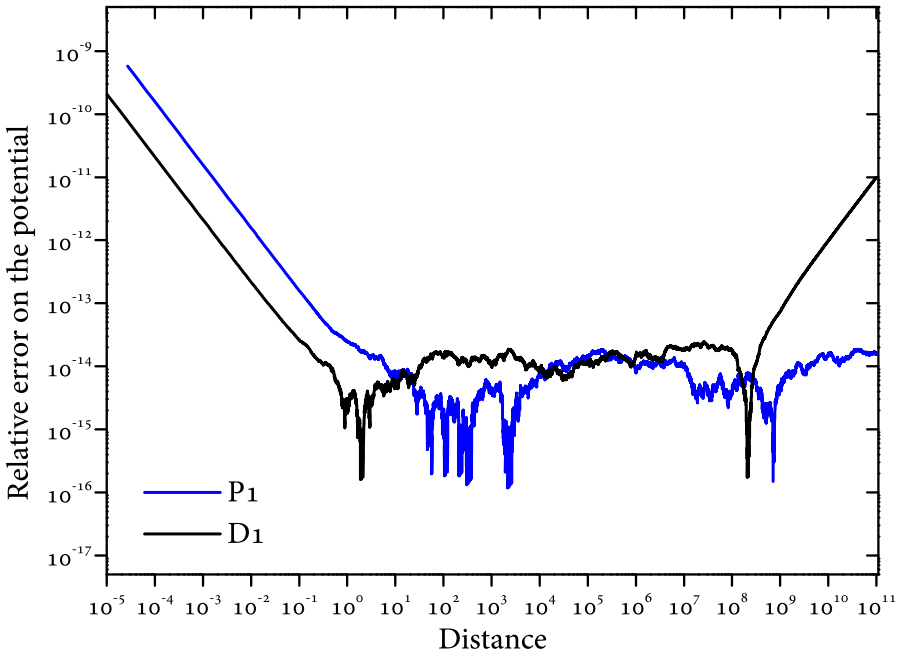


**Figure 4.3:** Relative errors on the numerical gravitational potential, shown in Fig. 4.2, for the tested models.

For completeness, we show in Fig. 4.2 the relative errors committed in evaluating the gravitational potential for the tested models `P1` and `D1`. It can be seen that the relative errors are quite good: always around $10^{-14}$ except in the last part of the integration (small distances) where, in any case, they remain satisfactory reaching the maximum value of $10^{-9}$ for $r \sim 10^{-5}$. We are currently trying to understand the reason why the relative errors tend to increase almost linearly, in a logarithmic scale, when $r \lesssim 1$ and, in the case of the system `D1`, for $r \gtrsim 10^9$ too. It is

important to underline that we need to integrate the Poisson's equation over a wide range of distances because when we will need to numerically solve equation (4.26), we will need to evaluate the quantity $\mathrm{d}\rho/\mathrm{d}\psi$ for $\psi_i \ll \psi_0$. In fact the integral in (4.26), for a certain value $\mathcal{E} = \widetilde{\mathcal{E}}$, has to be evaluated for values of $\psi$ such that $\psi \in \left[0, \widetilde{\mathcal{E}}\right]$ and values of $\psi$ very close to zero means always very large distances, except for the King model in which, indeed, the integration stops at $r \simeq r_t$. Therefore, it is also easy to understand why a good accuracy (thus a good integrator) is needed to solve the Poisson's equation.

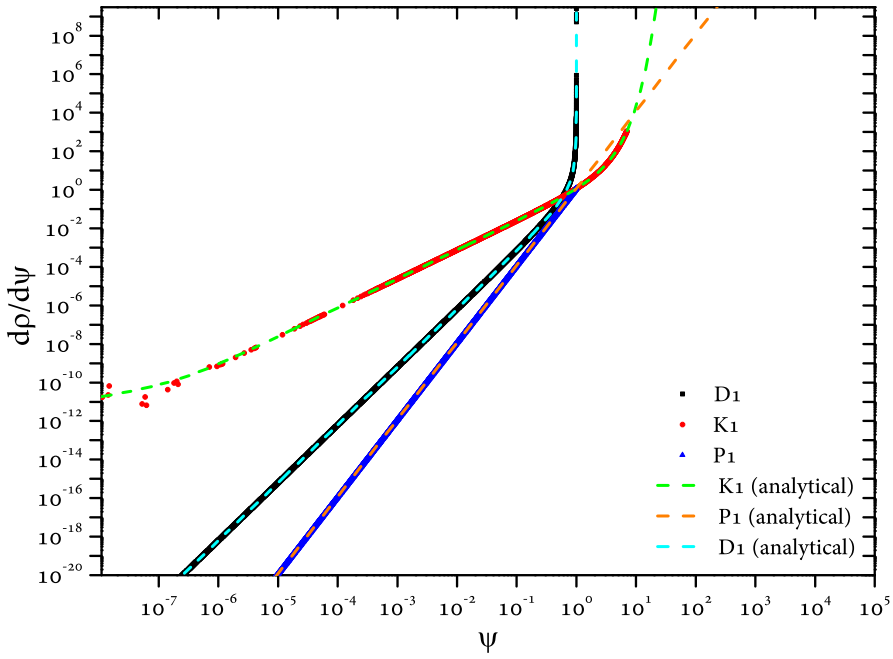


**Figure 4.4:** The derivative of the density with respect to the gravitational potential for the tested models, evaluated numerically using the procedure described in section 4.4.2.

Another important quantity which is shown in Fig. 4.4 is the numerical derivative $\frac{\mathrm{d}\rho}{\mathrm{d}\psi}$ calculated from the previously tabulated values of $\rho(r)$ and $\psi(r)$ using the method described in section 4.4.2. This is one of the most critical parts of the algorithm because it is based on a a quite rough estimation of the first derivative $\frac{\mathrm{d}\rho}{\mathrm{d}\psi}$. We know the analytical explicit formulas of $\frac{\mathrm{d}\rho}{\mathrm{d}\psi}$ for the systems `D1`, `K1` and `P1`; for the system

P1 the expression has already been obtained in section 4.1.2, equation (4.32) and it is

$$\frac{\mathrm{d}\rho_{\mathtt{P1}}(\psi)}{\mathrm{d}\psi} = \frac{15}{4\pi}\frac{b^2}{G^5M^4}\psi^4. \tag{4.64}$$

For the King model the density is given directly as a function of the dimensionless potential $W = \frac{\psi}{\sigma^2}$ (see equation 4.38). Changing variable from $W$ to $z \equiv \sqrt{W}$ we have

$$\frac{\rho_{K1}(z)}{\mathrm{d}z} = \left[2ze^{z^2}\mathrm{erf}(z) - \frac{4}{\sqrt{\pi}}z^2\right] \tag{4.65}$$

where we have used the property of the error function

$$\frac{\mathrm{d}}{\mathrm{d}z}\mathrm{erf}(z) = \frac{2}{\sqrt{\pi}}e^{-z^2}. \tag{4.66}$$

Coming back to the variable $W$ we get, except for a factor $1/\sigma^2$, the expression of $\frac{\mathrm{d}\rho}{\mathrm{d}\psi}$ for the King model

$$\frac{\mathrm{d}\rho_{K1}(W)}{\mathrm{d}W} = e^W\mathrm{erf}\left(\sqrt{W}\right) - \frac{2}{\sqrt{\pi}}\sqrt{W}. \tag{4.67}$$

For a generic Dehnen model it is possible to write (see for example [38])

$$\rho_{Dehnen}(\psi) = \frac{(3-\gamma)M}{4\pi a^3}\frac{(1-y)^4}{y^\gamma} \tag{4.68}$$

where we have introduced the dimensionless variable $y$ defined as

$$y \equiv \left[1 - (2-\gamma)\frac{a}{GM}\psi\right]^{\frac{1}{2-\gamma}}. \tag{4.69}$$

For the system D1, $\gamma = 1$, we have

$$\rho_{D1}(\psi) = \frac{M}{2\pi a^3}\left(\frac{a}{GM}\psi\right)^4\left(1 - \frac{a}{GM}\psi\right)^{-1}. \tag{4.70}$$

To calculate the derivative with respect to $\psi$ of equation 4.70 we introduce the dimensionless variable $x$ defined as

$$x \equiv \frac{a\psi}{GM} \tag{4.71}$$

in order to write

$$\rho_{D1}(x) = \frac{M}{2\pi a^3}\frac{x^4}{1-x}. \tag{4.72}$$

Therefore we can obtain

$$\frac{\mathrm{d}\rho_{D1}(\psi)}{\mathrm{d}\psi} = \frac{\mathrm{d}\rho_{D1}(x)}{\mathrm{d}x}\frac{\mathrm{d}x}{\mathrm{d}\psi} = \frac{1}{2\pi G a^2}\frac{4x^3 - 3x^4}{(1-x)^2}. \tag{4.73}$$

As we can see in Fig. 4.4, the exact expressions for $\frac{\mathrm{d}\rho}{\mathrm{d}\psi}$ (dashed lines) are not distinguishable, by eye, from the numerical points obtained using our code. The exception is constituted by the King model for the values of $\psi$ such that $\psi \lesssim 10^{-6}$. This may be due to, at least, three main reasons :

1. it is difficult to follow the rapid decrease of $\psi(r)$ (and $\rho(r)$) when the numerical resolution of the Poisson's equation is approaching to $r \sim r_t$ where, in particular, $\psi(r_t) = 0$. In particular, the integration step in the BS integrator must be reduced significantly and iteratively to follow, with enough accuracy, the trend of $\psi(r)$;

2. as we saw in section 4.1.3 the King's density profile is written in terms of the error function which must be evaluated numerically, adding, surely, another term of error (even if it is generally small compared to that described in point 1 and 3);

3. obviously, the method that we used to evaluate the numerical derivative is approximated and it suffers rapid relative variations of $\psi$ which occur at $r \sim r_t$, being based on Taylor expansions.

Especially the point 3 is crucial to obtain a sufficiently accurate `DF` . In fact, it can be seen in Fig.4.5 that, although, as we have seen previously, the errors on $\psi(r)$ are very small (see Fig.4.2), the errors on the quantity $\frac{\mathrm{d}\rho}{\mathrm{d}\psi}$ are significantly bigger. As already seen in Fig.4.3, in Fig,4.5 we find again that the error grows for $r \lesssim 1$ in each model as expected considering its natural propagation. The error for $r \ll 1$ in

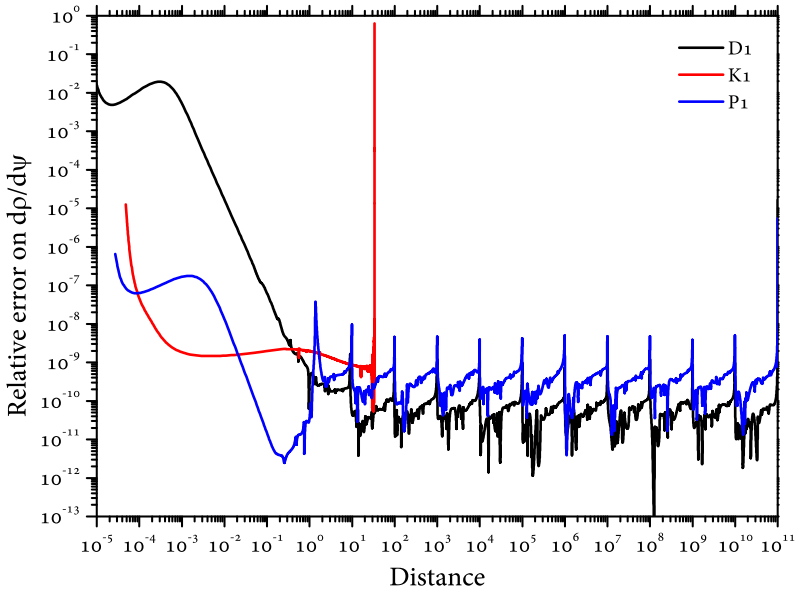


**Figure 4.5:** Relative errors on the derivative of the density with respect to the gravitational potential for the tested models.

the system `D1`, besides the already cited sources of error, comes from the difficulty to approximate, with enough accuracy, the divergence of $\frac{\mathrm{d}\rho_{D1}}{\mathrm{d}\psi}(\psi = 1; r = 0)$ (see equation 4.73). Moreover, as already seen, by eye, in Fig.4.5 it is evident the rapidly increasing trend of the error, for the model `K1`, when approaching the tidal radius ($r_t \simeq 37$ for $W_0 = 7$). Looking at the error for the models `P1` and `D1` it can be noticed a certain degree of periodicity which is not clear at present and it should be investigated deeper in order to reduce it (even if the oscillations remain around the value $10^{-10}$ which is, in any case, a very good error in $\frac{\mathrm{d}\rho}{\mathrm{d}\psi}$ ensuring the realization of a stable $N$-body system model).

Fig. 4.6 shows the `DF` , for the three tested models, resulting from the numerical integration of equation 4.26 and sampled on a grid composed by 5000 values of energies, for each curve, distributed logarithmically between $\mathcal{E}_{min}$ and $\mathcal{E}_{max}$. The dashed lines represent the analytical ex-

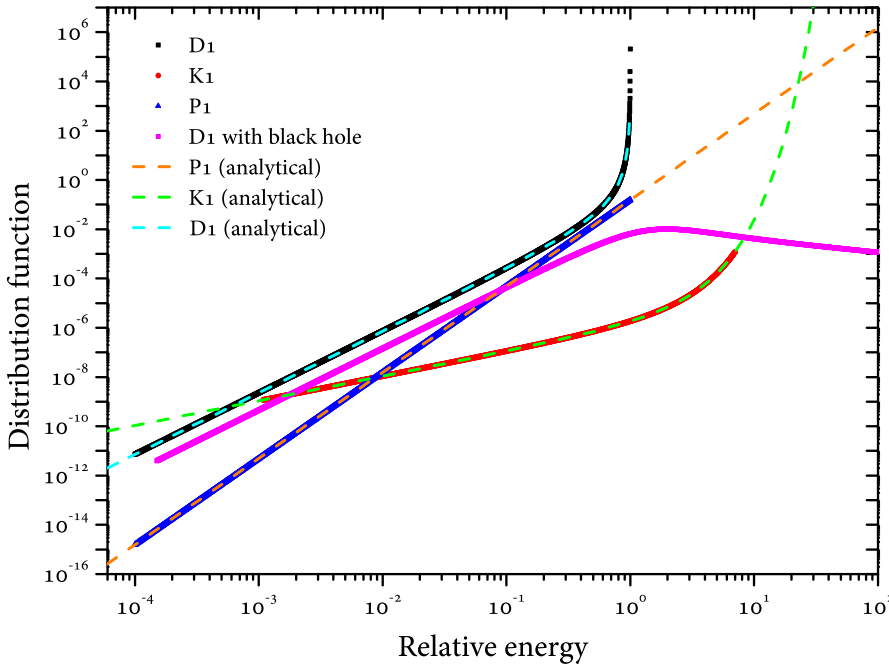


**Figure 4.6:** The distribution functions for the tested models obtained using our code. The distribution functions have been evaluated using a logarithmic grid in energies composed by 5000 points. The dashed lines represent the explicit, analytical forms of the distribution functions. The magenta points are referred to the system D1 in which a central super massive object, whose mass is equal to half the total system mass, is included.

pressions for the DF . No appreciable differences can be seen, by eye, between the dashed curves and the numerical evaluations and this result could be enough to guarantee sufficiently stable initial conditions for a $N$-body computer model. The explicit expression for the King DF and for the Plummer DF can be found respectively in equations 4.36 and 4.35 while it is possible to show that the DF for the model D1 is

$$\frac{1}{8\sqrt{2}\pi^3\alpha^3 G a^2}\frac{1}{\sqrt{\mathcal{E}}\left(\alpha-\mathcal{E}\right)^2}\left[-16E^4+24\alpha\mathcal{E}^3-2\alpha^2\mathcal{E}^2-3\alpha^3\mathcal{E}+3\alpha^4 K\mathrm{arctg}K\right] \tag{4.74}$$

where we have introduced $\alpha \equiv \frac{GM}{a}$ and $K \equiv \sqrt{\frac{\mathcal{E}}{\alpha-\mathcal{E}}}$. It is also evident that the DF , in each model, has a limited domain; in fact the DF does not

exist for all the values of $\mathcal{E}$ such that $\mathcal{E} > \psi(r=0) = \psi_{max}$. Since $\mathcal{E} \equiv \psi - \frac{1}{2}v^2 \Rightarrow \mathcal{E}^+_{lim} = \psi_{max}$. Considering that, for example, the King model `K1` has been represented in Fig. 4.6 using the dimensionless energy (that is scaled with a factor $\frac{1}{\sigma^2}$), we see that the right limit for the `DF` is $\mathcal{E}^+_{lim} = 7 = W_0$. The same can be said for the models `D1` and `P1` which are correctly limited by the value $\mathcal{E}^+_{lim} = 1$ which corresponds to an asymptote, as expected from equation 4.74, for the model `D1`. The left limit on the `DF` depends on the minimum value of $\psi$ ($\psi_{min}$) reached during the integration of the Poisson's equation. In fact, if, in the integral 4.26, we choose a value $\widetilde{\mathcal{E}} < \psi_{min}$ we cannot evaluate $f\left(\widetilde{\mathcal{E}}\right)$ because we do not have tabulated values of the integrand between 0 and $\widetilde{\mathcal{E}} = \psi_{min}$.

This is now clear definitively the importance to integrate Poisson's equation on a wide range in space. This implies that given a certain particle in position $r_i$, it cannot have an energy $\mathcal{E}_i \lesssim \mathcal{E}^-_{lim}$ therefore it cannot goes arbitrarily close to its local escape velocity which is, actually, not a big trouble for the initial conditions of a generic $N$-body system. For completeness, we also show in Fig. 4.6 (magenta points) a `D1` model which includes a central black hole with mass $M_{BH} = 0.5$, that is, half the total mass of the system. The `DF` relative to this system is not known explicitly but we can see, from our numerical results, how the presence of the central black hole has two main effects:

1. it removes the singularity of the `DF` of the model `D1` for $\mathcal{E} = \psi(r=0) = 1$;

2. it expands significantly the range of allowed energies for the resulting $N$-body model which now goes from $\sim 0$ to the value of the gravitational potential at the minimum distance reached by the integration (in our case it is fixed to $r_{min} = 10^{-3}$ therefore we have $\mathcal{E}^+_{lim} \simeq 500$).

The usage of "sampling " programs like ours is the only way to generate a $N$-body, stable, computer model containing a central super massive object. This is a very important point for astrophysicists because most astrophysical stellar systems, like for example most of galaxies, harbour a super massive black hole in their innermost regions.

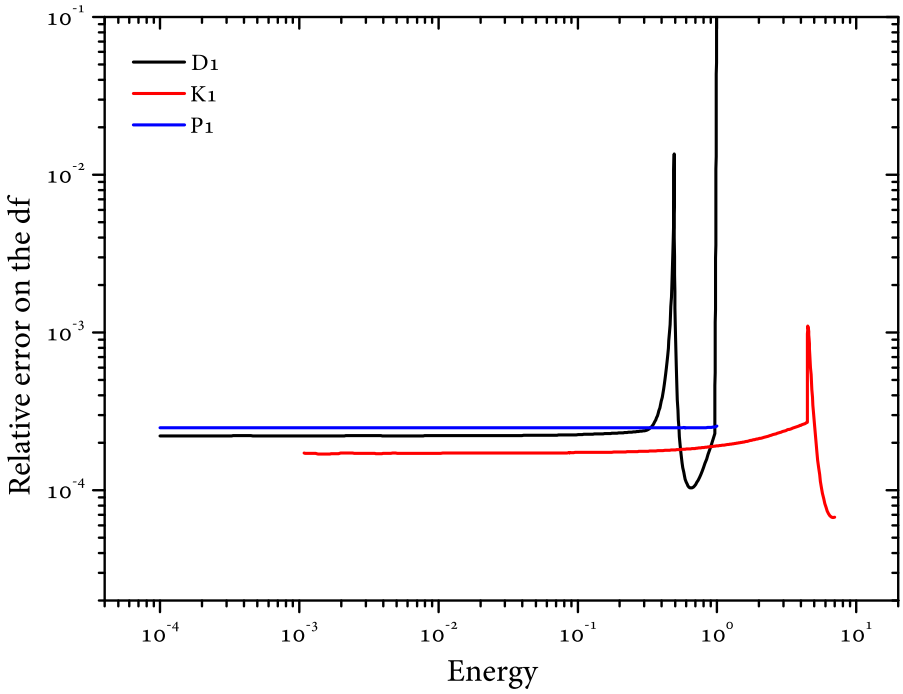


**Figure 4.7:** The relative errors on the distribution functions for the tested models.

To quantify more precisely the errors on the different DF of our models, we show in Fig.4.7 the relative errors on the DF . As we can see, they are approximatively constants around $10^{-4}$ which is due for the majority to the error done in evaluating the integral which appears in equation 4.26 which is, indeed, approximatively constant and very difficult to improve (we are currently working on it). In any case, we verified, and we will show some results in the next section, that this error on the DF guarantees a stable $N$-body system even considering the most critical situation which corresponds to the inclusion of a central super massive object.

In figures 4.8, 4.9, 4.10 we show a graphical representation (on the plane xy) of the resulting $N$-body system respectively for the models `D1`, `K1` and `P1` flanked by a plot which represents the associated phase space (velocity vs distance, both in absolute values, for each particle). Fig. 4.11 is the same of the previous 3 figures but it is referred to the system `D1` modified with the inclusion of a central super massive black hole. In each phase space diagram (except for the `K1` model) the analytical expression of the escape velocity (red line) is shown as a function of the distance from the centre of gravity of the considered systems. As we can see, in all the figures the particles are distributed, as expected, below the curve of the escape velocity (even in the case of the presence of a super massive black hole). This constitutes one more proof of the validity of our numerical implementation. In Fig. 4.11 it is worth noting how the black hole alters the phase space distribution of the stars, especially in the innermost regions of the system. The dashed blue line represents, in fact, the escape velocity from the original `D1` model (i.e. without the black hole) while the green dashed line is that from the central massive object only (excluding the Dehnen, $\gamma = 1$, gravitational potential). The red line represents the resulting escape velocity which is the sum of the two different contributions.

## 4.7 Stability tests

In order to verify, in practice, the stability of the sampled systems we chose to sample a system of practical astrophysical interest and to analyse the evolution of its mass density profile and of its lagrangian radii[3] over a certain interval of time. To dynamically evolve this system we used our direct summation $N$-body code `HiGPUs` already introduced, tested and discussed in the previous chapters. The system chosen for

[3]The radius which contains a certain percentage ($p$) of the total mass of the system is said to be the lagrangian radius of the $p$-percent of the total mass of the system.

this simulation is the system indexed as `C1`: it mimics the innermost region of a typical elliptical galaxy modelled using a Dehnen model with $\gamma = 0.2$ truncated using an hyperbolic secant function $\mathrm{sech}\left(\frac{r}{r_{cut}}\right)$ with $r_{cut} = 80\mathrm{pc}$ in order to obtain a good spatial resolution and quite realistic masses for the individual stars. The total mass of the galaxy is $M_g = 10^{11} M_\odot$ and we choose the value of the scale parameter $a$ using the reasonable condition $M\left(50\mathrm{kpc}\right) = 0.9 M_g$ which gives us the value $a \sim 1.9$ which corresponds to have a total mass of our truncated system around $8 \times 10^7 M_\odot$. We sampled this system using $N = 2^{20}$ stars including a central super massive black hole with mass $M_{BH} = 10^8 M_\odot$. We used an unit of length $R_s = 10\mathrm{pc}$ and an unit of mass $M_s = 10^4 M_\odot$ corresponding to a time unit of $\sim 4.7\mathrm{Myr}$. The crossing time for this system, considering a characteristic dimension $R_{sys} \simeq 100\mathrm{pc}$ and a total mass $M_{sys} \simeq 10^8 M_\odot$, is approximatively 1.5 Myr that is $\sim 0.3$ time units. The relaxation time is approximatively (using the formula 1.75) $\sim$ 2800 time units, that is $\sim$ 13 Gyr. We evolved this system for a sufficiently long time to verify the goodness of the sampled model (50 time units, that is $\sim$ 170 crossing times) using 4 AMD Radeon HD7970 to accelerate the simulation (see section 3.5).

Fig.4.12 shows the evolution of several lagrangian radii, normalized to their initial value for clarity of the representation, in function of time expressed in units of the system crossing time. We can see that, in the first 10 crossing times, the system does not suffer from rapid and significant variations of its initial state and, considering also the presence of a central black hole, this denotes the goodness of the generated computer model. The variations (less than 1%) seen during the first 10 crossing times are due to the fact that we used this galaxy model to study the dynamical evolution of a globular cluster in circular orbit embedded in the sampled environment. The variations seen in the initial values of the lagrangian radii are due to the perturbation induced by the globular clutser itself on the background. Obviously the long term evolution of the lagrangian radii reflects the tendence of the system to

evolve towards core collapse. Therefore the slightly decreasing trend of the curves relative to the inner systems regions and the simultaneous inverse trend of the other outermost lagrangian radii, denotes also the goodness of the $N$-body integrator (in this case our parallel code `HiGPUs` ). This is also confirmed by a value of the relative total energy variation of $\sim 10^{-9}$ after $\sim 170$ crossing times.

The same stability is observed by looking at Fig.4.13 which represents the initial (black line) and the final (red line) mass density profile of the system `C1`. It has been verified that wrong initial conditions produce rapid and violent changes of the density profile (over a time $\sim 10$ crossing times) which then stabilize on a curve with a significantly different shape. This not happens for our tested model (despite the presence of the central black hole) and the density profile remains approximatively untouched over $150$ crossing times. The appearance of the tail for the red line reflects the natural evolution of any collisional system: some stars acquire enough kinetic energy, thanks to close encounters, to get to large distances and, sometimes, to escape from the system. Simultaneously, in order to conserve the total energy, the system shrinks (which explains also the slightly increasing value of the final density in the central regions). In fact, the physical characteristics of the stellar systems sampled using the Boltzmann, collisionless, relation (see equation (4.5)) must not remain exactly the same because, in real $N$-body simulations, close encounters must be taken into account. Nevertheless, it is important to avoid rapid and violent initial variations of the initial conditions, in order to avoid spurious effect on the resulting scientific results. In real $N$-body simulations, the evolution of the `DF` , as we have already seen previously, follows the master equation

$$\frac{\mathrm{d}f}{\mathrm{d}t} = \Gamma\left[f\right] \tag{4.75}$$

which, nevertheless, cannot be solved explicitly in an exact way.

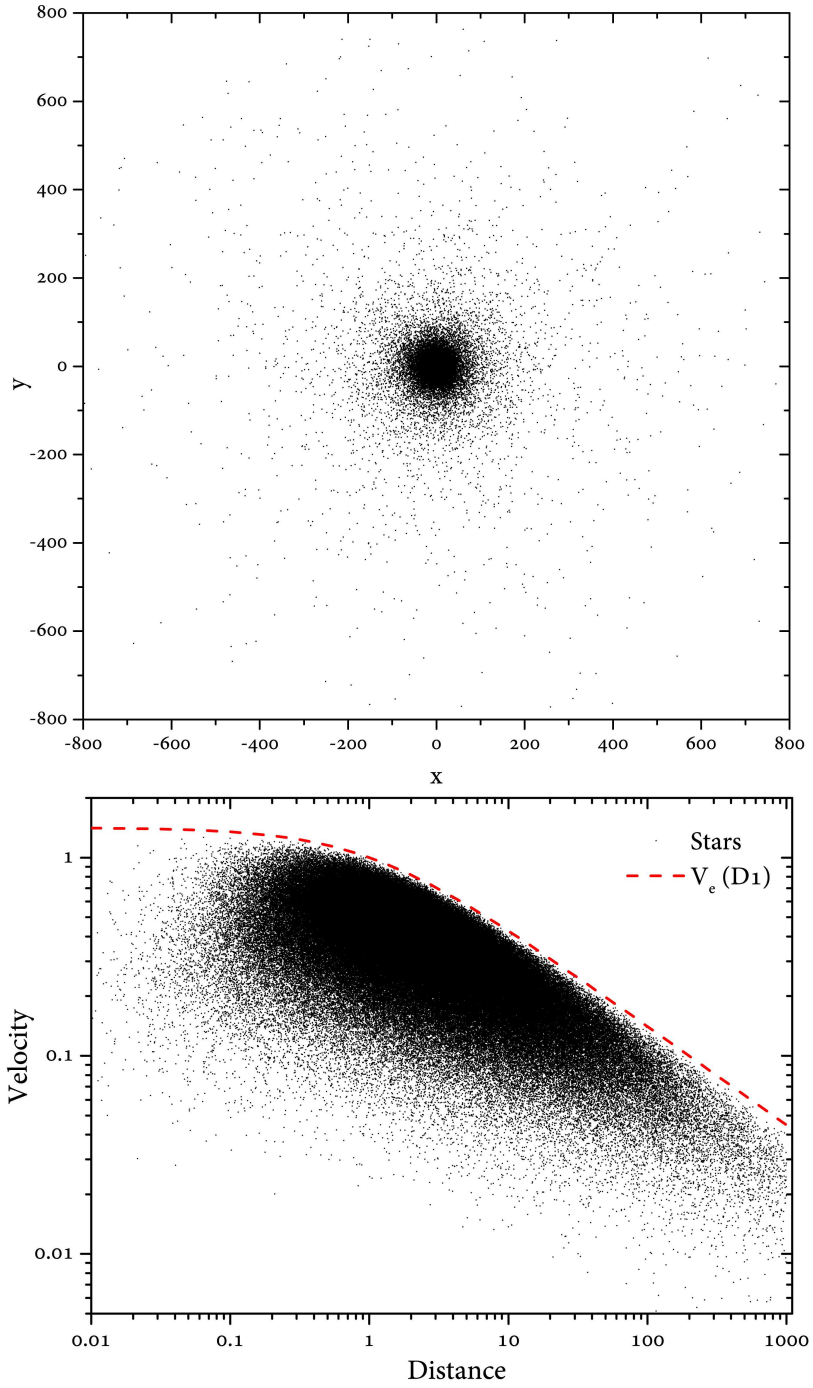


**Figure 4.8:** The top panel shows the $N$-body model, on the plane xy, resulting from the model `D1`. The bottom panel represents the phase space associated to the $N$-body model. The red line is the trend of the escape velocity in function of the distance.

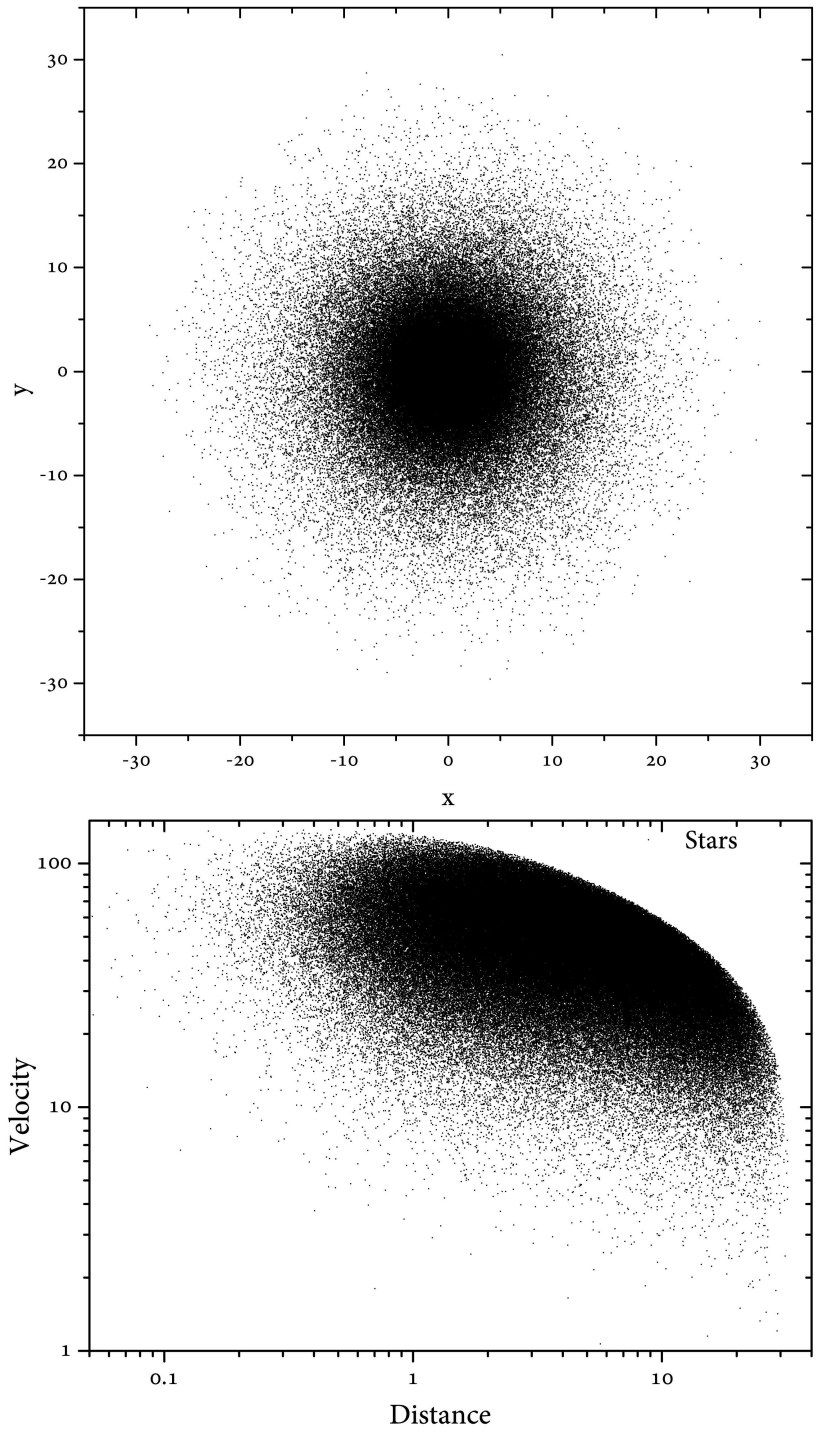


**Figure 4.9:** The top panel shows the $N$-body model, on the plane xy, resulting from the model `K1`. The bottom panel represents the phase space associated to the $N$-body model.

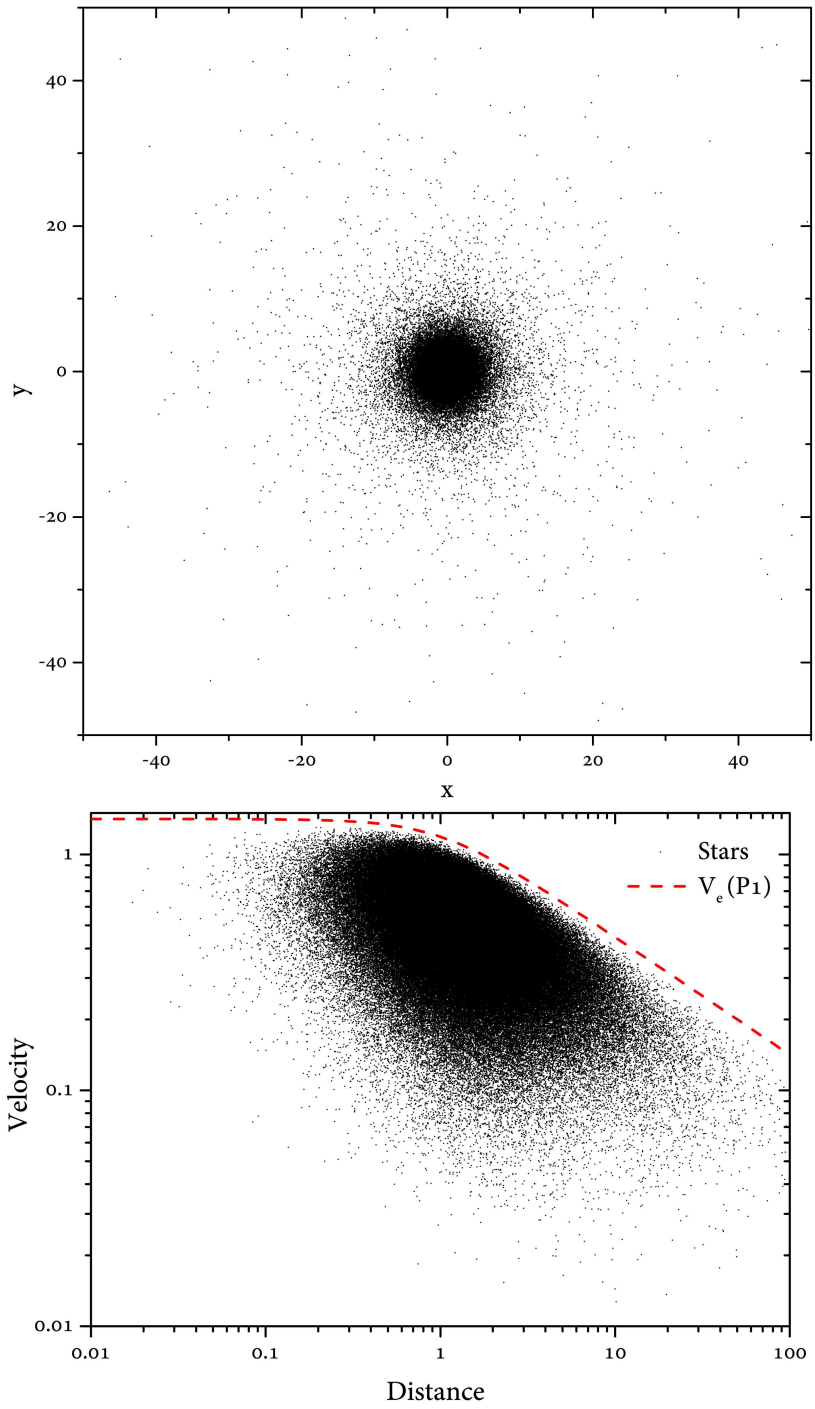


**Figure 4.10:** The top panel shows the $N$-body model, on the plane xy, resulting from the model `P1`. The bottom panel represents the phase space associated to the $N$-body model. The red line is the trend of the escape velocity in function of the distance.

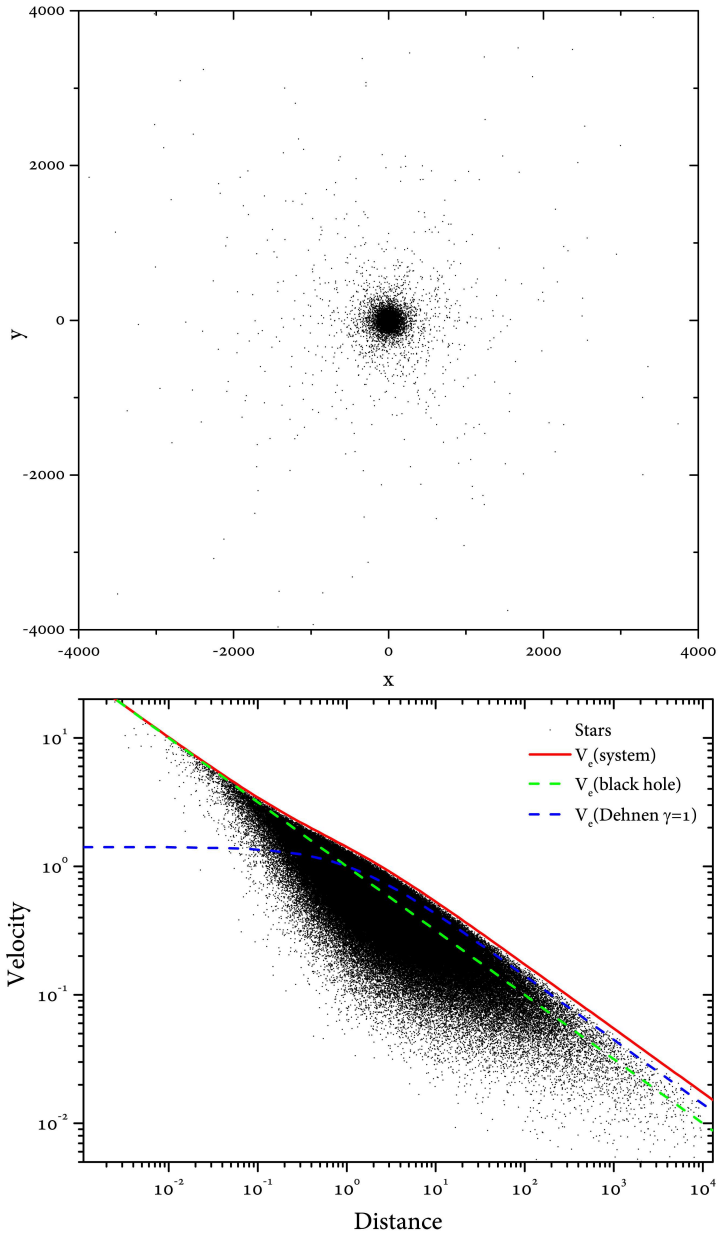


**Figure 4.11:** The top panel shows the $N$-body model, on the plane xy, resulting from the model `D1` modified with the inclusion of a central super massive particle. The bottom panel represents the phase space associated to the $N$-body model. The red line is the trend of the escape velocity in function of the distance while the blue dashed line is the escape velocity relative to the original `D1` model and the green dashed line represents the trend of the escape velocity considering the presence of the central black hole only.

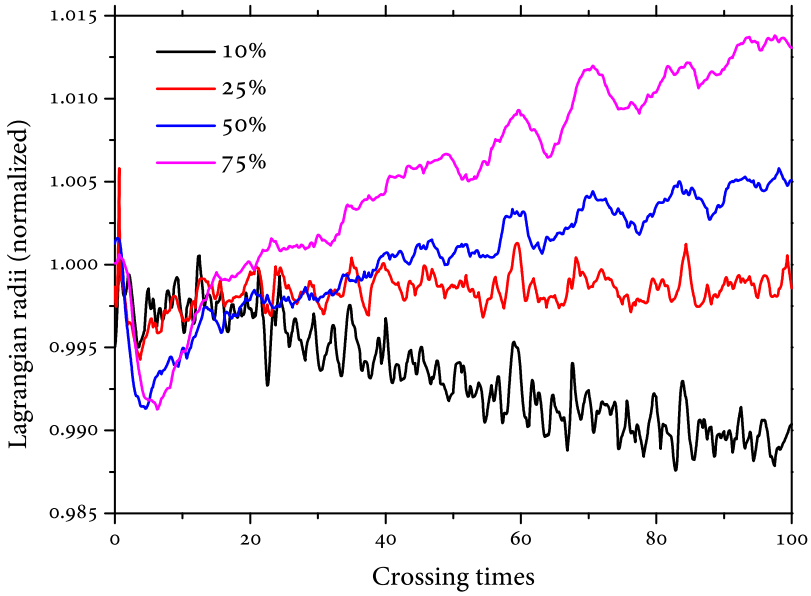


**Figure 4.12:** Lagrangian radii relative to different percentages of the total mass of the system `C1`. For convenience, each lagrangian radius has been normalized to its initial value.

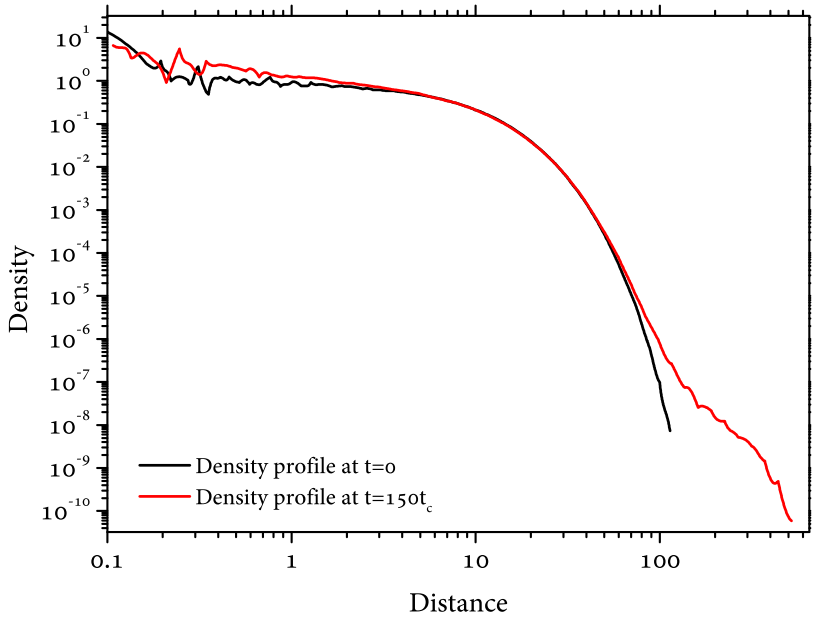


**Figure 4.13:** Mass density profile of the system `C1` at times $t = 0$ and $t = 170$ crossing times.

# 5 Regularization methods for the $N$-Body problem

## 5.1 Introduction

We have already seen in chapter 1 that, despite its simple mathematical formulation, which is substantially untouched since 1687, the numerical integration of the $N$-Body problem poses at least two problems:

1. the computational complexity of the problem is $O\left(N^2\right)$ because of the infra-red divergence of the gravitational potential. This implies that the times needed to complete a typical astrophysical simulation ($N \gtrsim 10^4$) are long;

2. because of the ultraviolet divergence, close encounters between stars, hard binaries and/or multiple systems are very difficult to integrate. In general, without using the softening parameter, the above listed situations become critical and the accuracy of the simulation is lost (even using a very small time step coupled with a high order integration algorithm).

We saw that modern hardware facilities like GPUs may help to overcome the first point but, besides the introduction of the softening parameter, we did not say anything about a possible solution which concerns the second point. In fact, the introduction of a smoothing factor is an useful artifice if we are interested to study the global properties of a certain $N$-Body system disregarding the small-scale phenomena such as the evolution of hard binaries or the precise reconstruction of the orbits

of stars. Actually there are, at least, three possibilities to overcome the ultraviolet divergence without introducing a smoothing parameter:

1. performing a smart coordinates transformation which can include, or not, the temporal variable;

2. choosing an algorithm which produces regular[1] results without changing coordinates;

3. the combination of the previous two points.

In general, any attempt to remove or to bypass the singularity of the 2-body interaction gravitational potential is referred as an attempt to regularize it, from which the world *regularization* derives.

## 5.2 The Burdet-Heggie regularization

We start describing a simple but efficient method, which transforms the temporal coordinate only, also known as the Burdet-Heggie (hereafter BH) method. The authors formulated independently this method and described it respectively in [21] and [51]. Let us consider, in a $N$-Body system, a binary star composed by one object of mass $m_i$ and its companion of mass $m_j$ posed in positions $\mathbf{r_i}$ and $\mathbf{r_j}$ with velocities $\mathbf{v_i}$ and $\mathbf{v_j}$ respectively. The equation of relative motion writes

$$\frac{\mathrm{d}^2\mathbf{R}}{\mathrm{d}t^2} = -\frac{GM}{R^3}\mathbf{R} + \mathbf{a_{ext}} \tag{5.1}$$

where we have introduced the relative distance $\mathbf{R} = \mathbf{r_i} - \mathbf{r_j}$, the total mass $M = m_i + m_j$ and the contribution to the acceleration due to the

[1] The word regular here means "not singular" where the singularity is that of the gravitational potential in $r = 0$

other $N-2$ stars ($\mathbf{a_{ext}}$). In order to eliminate the divergence of type $1/R^2$ we introduce the following differential time transformation

$$\mathrm{d}t = R^n \mathrm{d}\tau \tag{5.2}$$

where $n$ is a generic exponent which, in principle, can take arbitrary values. For convenience we denote the derivatives with respect to the new time coordinate $\tau$ using "primes" and those with respect to the old time $t$ with "dots" . Therefore, we have

$$\dot{\mathbf{R}} = \frac{\mathrm{d}\mathbf{R}}{\mathrm{d}\tau}\frac{\mathrm{d}\tau}{\mathrm{d}t} = \frac{1}{R^n}\mathbf{R}' \tag{5.3}$$

$$\ddot{\mathbf{R}} = \frac{\mathrm{d}\tau}{\mathrm{d}t}\frac{\mathrm{d}\dot{\mathbf{R}}}{\mathrm{d}\tau} = \frac{1}{R^{2n}}\mathbf{R}'' - n\frac{R'}{R^{2n+1}}\mathbf{R}'. \tag{5.4}$$

Substituting the expression for $\ddot{\mathbf{R}}$ in the equation of relative motion 5.1 we have

$$\mathbf{R}'' = \frac{nR'}{R}\mathbf{R}' - \frac{GM}{R^{3-2n}}\mathbf{R} + R^{2n}\mathbf{a_{ext}}. \tag{5.5}$$

It is worth noting that if we choose $n = 1$, we have already smoothed the singularity from $1/R^2$ to $1/R$, the latter surely better behaved for $R \to 0$. Using $n = 1$ the equation of motion becomes

$$\mathbf{R}'' = \frac{R'}{R}\mathbf{R}' - \frac{GM}{R}\mathbf{R} + R^2\mathbf{a_{ext}}. \tag{5.6}$$

We introduce now the *Laplace-Runge-Lenz* vector, $\mathbf{e}$, whose length is equal to the eccentricity of the orbit

$$\mathbf{e} \equiv \frac{\mathbf{V}\wedge\mathbf{C}}{GM} - \frac{\mathbf{R}}{R} = \frac{R'^2}{GMR^2}\mathbf{R} - \frac{R'}{GMR}\mathbf{R}' - \frac{\mathbf{R}}{R} \tag{5.7}$$

where $\mathbf{V} = \dot{\mathbf{R}} = \mathbf{v_i} - \mathbf{v_j}$ and $\mathbf{C} \equiv \mathbf{R}\wedge\mathbf{V}$. The introduction of the eccentricity let us write equation (5.6) in the following form

$$\mathbf{R}'' = 2\left(\frac{R'^2}{2R^2} - \frac{GM}{R}\right)\mathbf{R} - GM\mathbf{e} + R^2\mathbf{a_{ext}}. \tag{5.8}$$

We introduce also the total energy of the 2-body system

$$E_{2b} = \frac{1}{2}V^2 - \frac{GM}{R} = \frac{1}{2}\frac{R'^2}{R^2} - \frac{GM}{R} \tag{5.9}$$

therefore, using equation (5.9) to simplify equation (5.8), we obtain

$$\mathbf{R}'' = 2E_{2b}\mathbf{R} - GM\mathbf{e} + R^2\mathbf{a_{ext}} \tag{5.10}$$

which is the regularized equation of motion in which , in fact, the singularity at $R = 0$ has been completely removed. Obviously, equation (5.10) must be coupled with equations that describe the rates of change of $E_{2b}$, $\mathbf{e}$ and $t$ with respect to the new time coordinate $\tau$. These relations can be easily obtained deriving respect to $\tau$ the equations (5.7) and (5.9) and using also the definition (5.2)

$$\begin{cases} E'_{2b} = \mathbf{a_{ext}} \cdot \mathbf{R}' \\ \mathbf{e}' = 2\mathbf{R}\left(\mathbf{a_{ext}} \cdot \mathbf{R}'\right) - \mathbf{R}'\left(\mathbf{a_{ext}} \cdot \mathbf{R}\right) - \mathbf{a_{ext}}\left(\mathbf{R} \cdot \mathbf{R}'\right) \\ t' = R. \end{cases} \tag{5.11}$$

In the absence of the external perturbation $E_{2b}$ and $\mathbf{e}$ are constant and equation (5.10) becomes

$$\mathbf{R}'' = 2E_{2b}\mathbf{R} + \text{constant} \qquad (E_{2b} < 0) \tag{5.12}$$

that is a (regular) harmonic oscillator, subject to a constant force $GM\mathbf{e}$, which oscillates with a period

$$T_{BH} = \frac{\pi}{\sqrt{|E_{2b}|}}. \tag{5.13}$$

## 5.3 The Kustaanheimo-Stiefel regularization

Another method to regularize the $N$-Body problem, which operates transformations of both time and spatial coordinates is the Kustaanheimo-Stiefel (hereafter KS) method [61] whose formulation in 2 dimensions is due to Levi-Civita [62]. The idea of Levi-Civita was to introduce (beside the time transformation 5.2) the new variables $u_1$ and $u_2$ such that $R = u_1^2 + u_2^2$ and

$$\begin{cases} R_1 \equiv u_1^2 - u_2^2 \\ R_2 \equiv 2u_1 u_2. \end{cases} \tag{5.14}$$

The transformation may be rewritten in a more elegant and compact form introducing the so called Levi-Civita matrix $\mathcal{L}(\mathbf{u})$

$$\mathbf{R} = \mathcal{L}(\mathbf{u})\,\mathbf{u} \tag{5.15}$$

where

$$\mathcal{L}(\mathbf{u}) = \begin{bmatrix} u_1 & -u_2 \\ u_2 & u_1 \end{bmatrix}. \tag{5.16}$$

The expression for the regularized velocity can be obtained deriving the expression in 5.15

$$\dot{\mathbf{R}} = 2\mathcal{L}(\mathbf{u})\frac{\mathbf{u}'}{R}. \tag{5.17}$$

It is possible to show (see for example Aarseth [4]) that the regularized equation of motion in these coordinates can be written as

$$\mathbf{u}'' = \frac{1}{2}E_{2b}\mathbf{u} + \frac{1}{2}R\mathcal{L}^T(\mathbf{u})\,\mathbf{a_{ext}} \tag{5.18}$$

which must be coupled with

$$E'_{2b} = 2\mathbf{u}' \cdot \mathcal{L}^T(\mathbf{u})\,\mathbf{a_{ext}}. \tag{5.19}$$

In the absence of external perturbations we obtain, just like in the BH regularization, an harmonic oscillator with period

$$T_{KS_{2D}} = 2\sqrt{2}T_{BH}. \tag{5.20}$$

It is possible to show that a 3D generalization of relations 5.14 must involve the complex plane, nevertheless Kustaanheimo and Stiefel showed that a real 4D generalization can be achieved. The 4D expression of the Levi-Civita matrix is

$$\mathcal{L}(\mathbf{u}) = \begin{bmatrix} u_1 & -u_2 & -u_3 & u_4 \\ u_2 & u_1 & -u_4 & -u_3 \\ u_3 & u_4 & u_1 & u_2 \\ u_4 & -u_3 & u_2 & -u_1 \end{bmatrix}. \tag{5.21}$$

Equation 5.15 still holds but now $\mathbf{R} = (R_1, R_2, R_3, R_4)$ with

$$\begin{cases} R_1 = u_1^2 - u_2^2 - u_3^2 + u_4^2 \\ R_2 = 2(u_1u_2 - u_3u_4) \\ R_3 = 2(u_1u_3 + u_2u_4) \\ R_4 = 0 \end{cases} \tag{5.22}$$

and $R = u_1^2 + u_2^2 + u_3^2 + u_4^2$. Since we are applying the regularization method for, realistic, 3D systems, we have one degree of freedom for choosing the components of the four-vector $\mathbf{u} = (u_1, u_2, u_3, u_4)$. It is appropriate to choose $u_4 = 0 \Leftrightarrow R_1 > 0$ and $u_3 = 0 \Leftrightarrow R_1 < 0$ obtaining

$$R_1 > 0: \begin{cases} u_1 = \sqrt{\frac{1}{2}(R_1 + R)} \\ u_2 = \frac{1}{2}\frac{R_2}{u_1} \\ u_3 = \frac{1}{2}\frac{R_3}{u_1} \\ u_4 = 0 \end{cases} \qquad R_1 < 0: \begin{cases} u_1 = \frac{1}{2}\frac{R_2}{u_2} \\ u_2 = \sqrt{\frac{1}{2}(R - R_1)} \\ u_3 = 0 \\ u_4 = \frac{1}{2}\frac{R_3}{u_2} \end{cases} \tag{5.23}$$

in order to have the new coordinates $u_1$ and $u_2$ always well defined in the field of real numbers. The other relations, already obtained for the 2D case, still hold. To perform a simple, not perturbed, numerical KS test, the recipe is summarized in the following steps

1. transform cartesian in relative coordinates obtaining $\mathbf{R}$ and $\mathbf{V} = \dot{\mathbf{R}}$;

2. obtain KS coordinates $\mathbf{u}$ and $\mathbf{u}'$ applying respectively the relations 5.23 and their derivatives with respect to the new time coordinate $\tau$;

3. solve the equation of motion 5.18 with a standard algorithm for ordinary differential equations; actually, in the not perturbed case, the solution, as we have already shown, is an harmonic oscillator; at the end we evolved $\mathbf{u}$ and $\mathbf{u}'$ from regularized time $\tau_0$ to $\tau_0 + \Delta\tau$;

4. transform the regularized coordinates to cartesian one using relations 5.22 and their derivatives;

5. obtain the physical time using the definition 5.2;

6. add the centre of mass motion to recover original (physical) positions and velocities.

## 5.4 Generalization to $N$ bodies

### 5.4.1 The Chain treatment

The KS formulas are valid for the perturbed 2-body problem and remain applicable even if the perturbation becomes quite strong. Nevertheless, during the integration of a $N$-Body system, a close encounter which involves, for example, two hard binaries, can happen, therefore it is needed to apply KS transformations not only to two bodies but to the entire (small) $N$-Body system. Unfortunately, the regularized equation of motion does not represent a simple harmonic oscillator any more although several "global" methods, which let us still regularize all the close approaches with $N > 2$, exist. We focus our attention on the description of the so called *chain treatment* which was introduced and subsequently improved by Seppo Mikkola and Sverre Aarseth [72]. To build the chain, first of all, the shortest inter-particle vector must be identified; this constitutes the first segment of the chain. Next, the closest particle to one or the other extreme is added to the chain, and so on, until all the stars to regularize are included. If the index $k$ refers to the ordered particles which belong to the chain we introduce the new chain vectors defined as

$$\mathbf{X}_k = \mathbf{r}_{k+1} - \mathbf{r}_k \qquad \mathbf{V}_k = \mathbf{v}_{k+1} - \mathbf{v}_k. \tag{5.24}$$

To go back to physical coordinates it is sufficient to assign to the first particle $\mathbf{r}_1 = \mathbf{0}$ and $\mathbf{v}_1 = \mathbf{0}$, apply recursively the definitions (5.24) and then rescale with respect to the centre of mass position and velocity. The chain method is not only fundamental and elegant to apply, for example, KS transformations to a generic $N$-Body system but also reduces significantly round-off errors especially the already discussed problem of numerical terms cancellation. The main disadvantage is that the chain structure can change during the dynamical evolution therefore,

it must be inspected every time step and, eventually, updated. In this framework the advantage is that it is not needed to go back to physical coordinates and rebuild the new chain but it is possible to show that if the positions of the $k$-th and $j$-th chained particles in the old chain are $I_k^{old} = I_\mu^{new}$ and $I_j^{old} = I_{\mu+1}^{new}$ we have

$$\mathbf{X}_\mu^{new} = \sum_{\nu=1}^{m-1} B_{\mu\nu} \mathbf{X}_\nu^{old} \tag{5.25}$$

where $m$ is the number of the particles to regularize and

$$B_{\mu\nu} = \begin{cases} +1 & k \leq \nu \cup j > \nu \\ -1 & j \leq \nu \cup k > \nu \\ 0 & \text{otherwise.} \end{cases} \tag{5.26}$$

To apply KS transformations to a $N$-body system we need to find the new equation of motion. First of all we need to express the hamiltonian in chain coordinates, then we need to switch to KS coordinates and perform the related time transformation. When dealing with a $N$-Body system, equation 5.2 must be generalized in the following form

$$t' = \frac{1}{L} = \frac{1}{T+U} \tag{5.27}$$

where $L$ is the Lagrangian, $T$ the kinetic energy and $U$ the gravitational potential energy of the system. The process to obtain the explicit form of the new equations of motion (regularized Hamilton-Jacobi equations) is laborious and the final result is also quite difficult to implement numerically. Anyway, the entire detailed analytical procedure can be found in Aarseth [4]. Surely KS regularization is a very powerful tool which, nevertheless, mainly because of the spatial coordinates transformation, is quite difficult to implement if coupled with the discussed chain treatment.

### 5.4.2 The Mikkola's Algorithmic Regularization

We now concentrate our attention on a valid alternative based on a combination of a time-only transformation (whose power has already been shown for BH and KS regularization) with a leapfrog algorithm which produces regular results despite the singularity in the mutual force. Seppo Mikkola can be considered the father of the so called *algorithmic regularization*. He developed and tested it completely, for the first time, with the help of Tanikawa in 1999 [73] and independently from Preto and Tremaine [85]. Several anecdotal about the invention of the algorithmic regularization and its development can be found in the interesting section 3 of the paper by Mikkola [71]. A generic step of the (symplectic) leapfrog algorithm, used to advance from time $t_0$ to $t_1 = t_0 + h$ the position $\mathbf{r}$ and velocity $\mathbf{v}$ of a generic particle which suffers an acceleration $\mathbf{a}(\mathbf{r})$, writes

$$\begin{cases} \mathbf{v_1} = \mathbf{v_0} + h * \mathbf{a}\left(\mathbf{r_{1/2}}\right) \\ \mathbf{r_1} = \mathbf{r_{1/2}} + \frac{h}{2}\mathbf{v_1}. \end{cases} \tag{5.28}$$

This scheme is not self-starting because we need to evaluate the quantity $\mathbf{r_{1/2}} = \mathbf{r_0} + \frac{h}{2}\mathbf{v_0}$. This algorithm has been proven to be very powerful, especially if coupled with variable, symmetrized, time steps, to integrate critical situations like close encounters producing regular results without using time or spatial coordinates transformations (see Fig. 5.1). On the other hand, we saw how time transformations can manipulate the Hamiltonian in order to completely remove the singularity in $R = 0$ (or, at least, to smooth it). The combination of these two strategies constitutes the base of the algorithmic regularization (hereafter AR). Let us consider a generic time transformation (from variable $t$ to variable $s$) such that

$$\mathrm{d}t = g(\mathbf{q}, \mathbf{p}, t)\,\mathrm{d}s \tag{5.29}$$

where we have used the Hamilton-Jacobi formalism indicating the coordinates with $\mathbf{q}$ and their conjugate momenta with $\mathbf{p}$. The Hamiltonian $\Gamma$, in the new coordinates, can be obtained considering the *extended Hamiltonian* in the phase space $\tilde{H}$ which is

$$\tilde{H}(\mathbf{q}, \mathbf{p}, t) = B + H(\mathbf{q}, \mathbf{p}, t) \tag{5.30}$$

where $B \equiv -H(\mathbf{q}(0), \mathbf{p}(0), t = 0)$. It is possible to show that the new Hamiltonian is

$$\Gamma = g(\mathbf{q}, \mathbf{p}, t)(B + H(\mathbf{q}, \mathbf{p}, t)). \tag{5.31}$$

In order to use a leapfrog algorithm we must have a separable Hamiltonian which allows the right hands of the equations of motion to be independent from the left sides. Nevertheless, in principle, the general form 5.31 is not separable in fact the equations of motion in the new coordinates write

$$t' = \frac{\partial \Gamma}{\partial B} = g \qquad \mathbf{q}' = \frac{\partial \Gamma}{\partial \mathbf{p}} = g\frac{\partial H}{\partial \mathbf{p}}$$
$$B' = -\frac{\partial \Gamma}{\partial t} = -g\frac{H}{t} \qquad \mathbf{p}' = -\frac{\partial \Gamma}{\partial \mathbf{q}} = -g\frac{\partial H}{\partial \mathbf{q}}$$

where primes indicate the derivatives with respect to the new variable $s$ (often called *regularized time*) and we have considered that $B + H = 0$ along the correct solution.

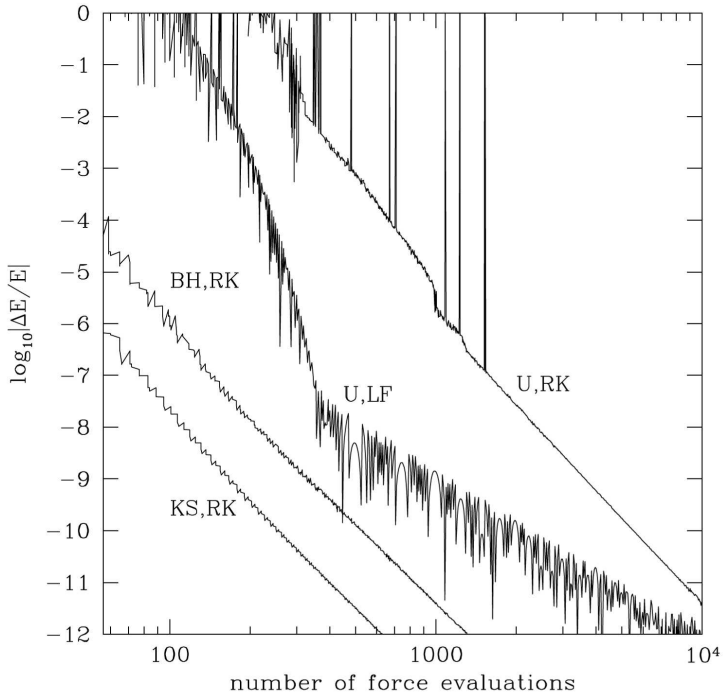


**Figure 5.1:** This figure is taken from Binney and Tremaine [18] section 3.4.7. It represents the relative energy variation obtained integrating one pericentre passage of a highly eccentric orbit in a Keplerian potential, as a function of the number of force evaluations. The orbit has semi-major axis $a = 1$ and eccentricity $e = 0.99$. Curves labeled by RK are followed using a fourth-order Runge-Kutta integrator with adaptive timestep control. The word U stands for unregularized, the curve BH uses Burdet-Heggie regularization, and KS means Kustaanheimo-Stiefel regularization. The curve labeled U,LF is followed in Cartesian coordinates using a leapfrog integrator 5.28 with variable timestep. The horizontal axis is the number of force evaluations used in the integration. The leapfrog method shows its validity being less precise of regularized codes but significantly more efficient (although it is only second order accurate) then the widely used RK (4th order) scheme. In this sense the leapfrog algorithm 5.28 is said to produce "regular" results.

### Time Transformed Leapfrog (TTL)

Let us consider the $k$-th particle in a $N$-Body system with position $\mathbf{r}_k$, velocity $\mathbf{v}_k$ and acceleration $\mathbf{a}_k$ at regularized time $s = s_0$. If we choose the function to transform time such as

$$g = \frac{1}{\Omega(\mathbf{r}_i)} \qquad \Omega(\mathbf{r}_i) > 0 \text{ and } i = 1, 2 \ldots, N \tag{5.32}$$

with $\Omega(\mathbf{r}_i)$ completely arbitrary, we will have, for example,

$$\mathbf{r}'_k = \frac{1}{\Omega(\mathbf{r}_i)} \frac{\partial H}{\partial \mathbf{p}_i} \tag{5.33}$$

which does not allow us to apply the leapfrog algorithm (5.28). Nevertheless, we can introduce a new auxiliary quantity $W = \Omega$ but, instead to think this new value such that $W = W(\mathbf{r}_i) = \Omega(\mathbf{r}_i)$, we consider $W$ as a variable which evolves numerically following the differential equation

$$\dot{W} = \sum_{i=1}^{N} \mathbf{v}_i \cdot \frac{\partial \Omega}{\partial \mathbf{r}_i} \quad \text{or} \quad W' = \frac{1}{\Omega} \sum_{i=1}^{N} \mathbf{v}_i \cdot \frac{\partial \Omega}{\partial \mathbf{r}_i}. \tag{5.34}$$

This allow us to separate the equations of motion of the generic particle $k$ in two different systems

$$\begin{cases} \mathbf{r}'_k = \frac{1}{W}\mathbf{v}_k \\ t' = \frac{1}{W} \\ \mathbf{v}'_k = \mathbf{0} \\ W' = 0 \end{cases} \qquad \begin{cases} \mathbf{v}'_k = \frac{1}{\Omega}\mathbf{a}_k \\ W' = \frac{1}{\Omega}\sum_{i=1}^{N} \mathbf{v}_i \cdot \frac{\partial \Omega}{\partial \mathbf{r}_i} \\ \mathbf{r}'_k = \mathbf{0} \\ t' = 0 \end{cases} . \tag{5.35}$$

In the following we will use the intuitive subscripts $0$, $1$ and $1/2$ to indicate quantities calculated respectively at $s = s_0$, $s_1 = s_0 + \Delta s$ and $s_{1/2} = s_0 + \frac{\Delta s}{2}$. Moreover, we use the expression of the acceleration of the $k$-th particle of the chain

$$\mathbf{A}_k = -\sum_{k\neq j}^{N} m_j \frac{\mathbf{r}_{kj}}{r_{kj}^3}. \tag{5.36}$$

Introducing now the chain coordinates $\mathbf{X}_k$ and $\mathbf{V}_k$, systems (5.35) can be rewritten in a compact form using a generic leapfrog step which evolves chained positions and velocities from (regularized) time $s = s_0$ to $s_1 = s_0 + \Delta s$ passing through $s_{1/2} = s_0 + \frac{\Delta s}{2}$

$$\mathcal{R}_k\left(\frac{\Delta s}{2}\right) \quad : \quad \begin{cases} \mathbf{X}_{k_{1/2}} = \mathbf{X}_{k_0} + \frac{\Delta s}{2W_0}\mathbf{V}_{k_0} \\ t_{1/2} = t_0 + \frac{\Delta s}{2W_0} \end{cases} \tag{5.37}$$

$$\mathcal{V}_k\left(\Delta s\right) \quad : \quad \begin{cases} \mathbf{V}_{k_1} = \mathbf{V}_{k_0} + \frac{\Delta s}{\Omega_{1/2}}\left(\mathbf{A}_{k+1} - \mathbf{A}_k\right)_{1/2} \\ W_1 = W_0 + \frac{\Delta s}{\Omega_{1/2}} \sum_{i=1}^{N} \frac{\partial \Omega}{\partial \mathbf{r}_i}_{1/2} \cdot \mathbf{v}_{i_{1/2}} \end{cases} \tag{5.38}$$

$$\mathcal{R}_k\left(\frac{\Delta s}{2}\right) \quad : \quad \begin{cases} \mathbf{X}_{k_1} = \mathbf{X}_{k_{1/2}} + \frac{\Delta s}{2W_1}\mathbf{V}_{k_1} \\ t_1 = t_{1/2} + \frac{\Delta s}{2W_1} \end{cases} \tag{5.39}$$

The quantity $\mathbf{v}_{i_{1/2}}$ is not known but we can approximate it averaging the values of $\mathbf{v}_{i_0}$ and $\mathbf{v}_{i_1}$. If we want to evolve the system from time $s_0$ to time $s_1$ performing $n$ iterations of step $\Delta s$, we can write the leapfrog algorithm as

$$\mathcal{R}_k\left(\frac{\Delta s}{2}\right)\left[\prod_{\nu=1}^{n-1}\left(\mathcal{V}_k\left(\Delta s\right)\mathcal{R}_k\left(\Delta s\right)\right)\right]\mathcal{V}_k\left(\Delta s\right)\mathcal{R}_k\left(\frac{\Delta s}{2}\right). \tag{5.40}$$

The algorithm illustrated in (5.39) is known with the name of *Time Transformed Leapfrog* or simply TTL. The recipe for the function $\Omega\left(\mathbf{r}\right)$ is given by many numerical experiments

$$\Omega\left(\mathbf{r}\right) = \sum_{i<j}^{N} \frac{\Omega_{ij}}{r_{ij}} \quad \text{where} \quad \Omega_{ij} = \begin{cases} \tilde{m}^2 \text{ if } m_i m_j < \epsilon \tilde{m}^2 \\ 0 \text{ otherwise} \end{cases} \tag{5.41}$$

with

$$\tilde{m}^2 = \sum_{i<j}^{N} \frac{2 m_i m_j}{N\left(N-1\right)} \tag{5.42}$$

and $\epsilon \sim 10^{-3}$ may be a good guess. It is worth noting that the physical time obtained from the regularized variable $s$ using the equation for $t'$ in the generic step $\mathcal{R}_k(\Delta s)$ is not correct and should be changed using its definition 5.29, that is

$$\int_{t_0}^{t_1=t_0+\Delta t} \mathrm{d}t = \int_{s_0}^{s_1=s_0+\Delta s} \frac{\mathrm{d}s}{W} \Rightarrow t_1 = t_0 + \int_{s_0}^{s_1} \frac{\mathrm{d}s}{W} \tag{5.43}$$

but the quantity $\frac{1}{W}$ cannot be taken out of the integral because $W = W(s)$. An approximated solution can be obtained solving the integral using the trapezoidal rule

$$\Delta t = \int_{s_0}^{s_1} \frac{\mathrm{d}s}{W} \simeq \frac{\Delta s}{2}\left[\frac{1}{W_1} + \frac{1}{W_0}\right] + O(\Delta s)^3 \tag{5.44}$$

but this implies that this kind of method produces a time (phase) error of $O(\Delta s^3)$.

### The Logarithmic Hamiltonian

If we choose the function

$$g = \frac{1}{U} \tag{5.45}$$

to transform the time coordinate, where $U$ is the total potential energy of the system to regularize, we obtain the new extended hamiltonian

$$\Gamma = \frac{T - U + B}{U} \tag{5.46}$$

where T is the total kinetic energy. $\Gamma = 0$ on the correct solution therefore $T + B = U$ letting us write

$$g = \frac{1}{T + B} = \frac{1}{U}. \tag{5.47}$$

In this case, system (5.39) can be written as

$$\mathcal{R}_k\left(\frac{\Delta s}{2}\right) \quad : \quad \begin{cases} \mathbf{X}_{k_{1/2}} = \mathbf{X}_{k_0} + \frac{\Delta s}{2(T+B)_0}\mathbf{V}_{k_0} \\ t_{1/2} = t_0 + \frac{\Delta s}{2(T+B)_0} \end{cases} \tag{5.48}$$

$$\mathcal{V}_k\left(\Delta s\right) \quad : \quad \begin{cases} \mathbf{V}_{k_1} = \mathbf{V}_{k_0} + \frac{\Delta s}{U_{1/2}}\left(\mathbf{A}_{k+1} - \mathbf{A}_k\right)_{1/2} \\ B_1 = B_0 + \frac{\Delta s}{U}\sum_{i=1}^{N}\left(-m_i \mathbf{v}i_{1/2} \cdot \mathbf{f}_{k_{1/2}}\right) \end{cases} \tag{5.49}$$

$$\mathcal{R}_k\left(\frac{\Delta s}{2}\right) \quad : \quad \begin{cases} \mathbf{X}_{k_1} = \mathbf{X}_{k_{1/2}} + \frac{\Delta s}{2(T+B)_1}\mathbf{V}_{k_1} \\ t_1 = t_{1/2} + \frac{\Delta s}{2(T+B)_1} \end{cases} \tag{5.50}$$

where $\mathbf{f}$ takes into account the presence of an external perturbation. This method is called the *logarithmic hamiltonian* algorithm or simply `LogH`. In fact, a functional (logarithmic) manipulation of the hamiltonian (5.46) allows us to write the hamiltonian $\Gamma$ in its separable form

$$\Lambda = \log\left(T + B\right) - \log U \tag{5.51}$$

to whom the leapfrog algorithm can be applied without problems. The same observations pointed out about the time error of the `TTl` method still hold. Actually the `LogH`, the `TTL` and the standard leapfrog method (no time transformation) can be collected using the generalized transforming function

$$g = \frac{1}{\alpha\left(T + B\right) + \beta\Omega + \gamma} = \frac{1}{\alpha U + \beta W + \gamma} \tag{5.52}$$

which is a function of the three parameters $(\alpha, \beta, \gamma)$. The combination $(1, 0, 0)$ is equivalent to the `LogH` method, $(0, 1, 0)$ is the `TTL` and $(0, 0, 1)$ is the standard leapfrog scheme. The ideal combination of parameter $(\alpha, \beta, \gamma)$ must be determined through numerical experiments. We must say that, in any case, the described schemes are only second order accurate and must be coupled with a powerful method to extrapolate or interpolate numerical results like, for example, the `BS` integrator (see section 4.4.1). This yields the complexity of the regularization (in terms

of both difficulty of implementation and operations to be executed) to very high levels.

### 5.4.3 Our implementation and tests

We now show some brief tests in order to demonstrate the numerical accuracy of the regularization. We implemented the general form of the Mikkola's algorithmic regularization (5.52) thanks also to the precious suggestions of Seppo Mikkola himself and we compare the results with those obtained using a CPU version of the widely used Hermite's 4th order integrator implemented using Block Time Steps (BTS). The first test we performed is referred to a system composed by two bodies with masses $m$ and $M$ with $M \gg m$. We ran several simulations varying the eccentricity of the orbit. We know that, for the 2-Body problem, we have [2]

$$R_a = \frac{1}{G\left(M+m\right)} \frac{R_a^2 V_a^2}{1-e} \Rightarrow e = 1 - \frac{R_a}{G\left(M+m\right)} V_a^2 = 1 - \frac{V_a^2}{V_{circ}^2} \quad (5.53)$$

where $e$ is the eccentricity, $R_a$ and $V_a$ respectively the relative position and velocity at the apocentre and $V_{circ}$ the relative velocity which corresponds to have a circular orbit. If we assign a relative velocity at apocentre $V_a = \zeta V_{circ}$ we have $e = 1 - \zeta^2$. We performed tests using $\zeta = 1 (e = 0)$, $\zeta = 0.5 (e = 0.75)$, $\zeta = 0.2$ $(e = 0.96)$ and the most critical case $\zeta = 0.01$ $(e = 0.9999)$.

[2] It is worth remembering that the 2-Body problem is completely equivalent to considering the motion of a test particle attracted by a fixed centre with mass equal two the total mass of the system.

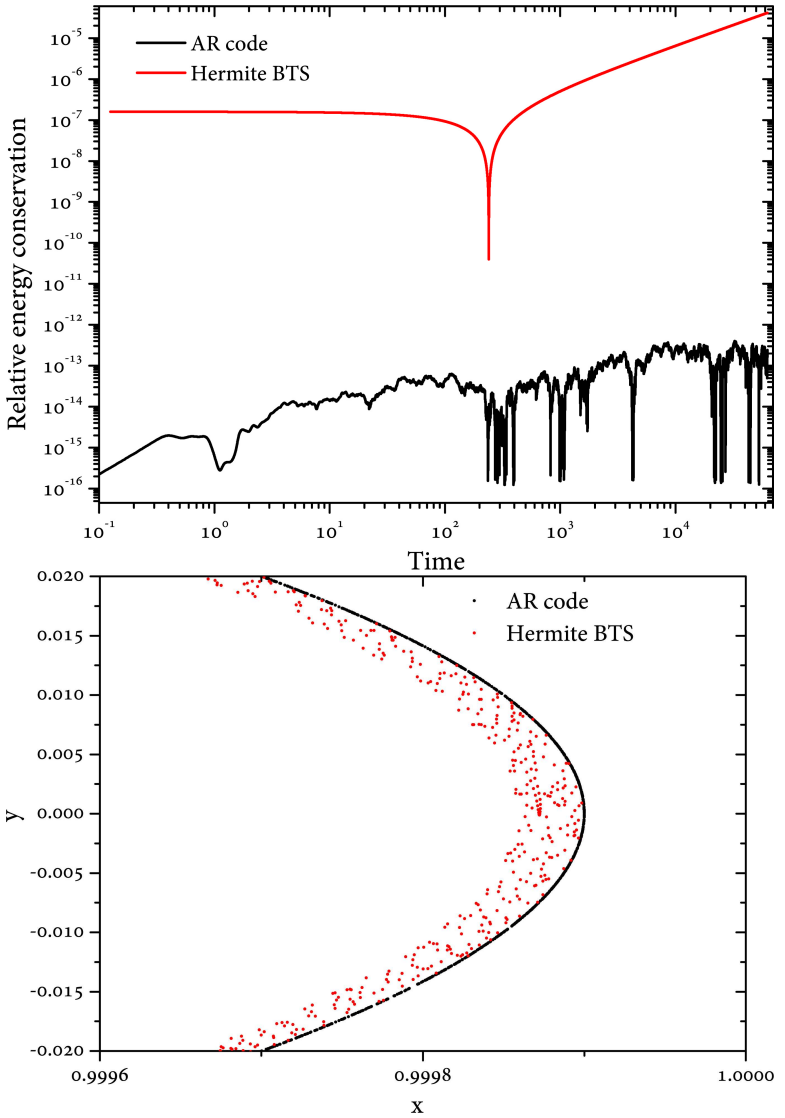


**Figure 5.2:** Case of eccentricity $e = 0$. Relative errors in total energy (top panel) for a standard (red) an regularized (black) algorithm. The bottom panel shows the positions (around apocentre) obtained by the standard (red) and regularized (black) integrators evolving the system over a time interval corresponding to $\sim 10^4$ orbital revolutions of the lighter star around the more massive particle.

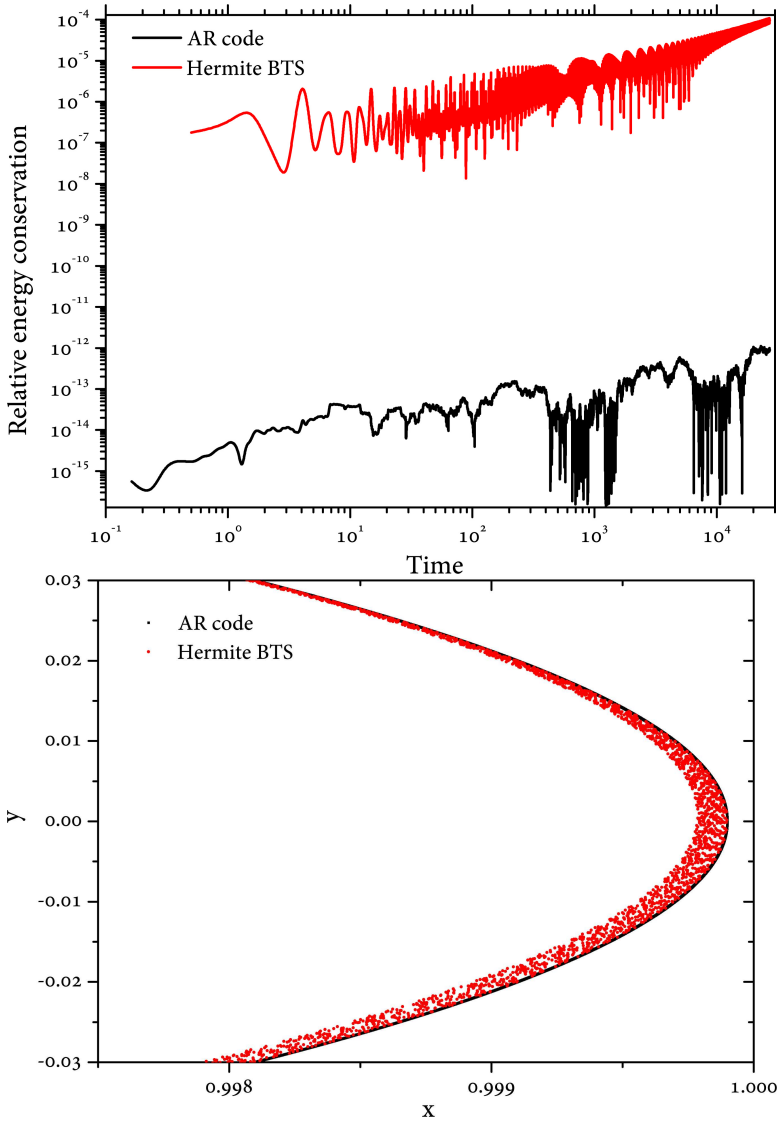


**Figure 5.3:** Case of eccentricity $e = 0.75$. Relative errors in total energy (top panel) for a standard (red) an regularized (black) algorithm. The bottom panel shows the positions (around apocentre) obtained by the standard (red) and regularized (black) integrators evolving the system over a time interval corresponding to $\sim 10^4$ orbital revolutions of the lighter star around the more massive particle.

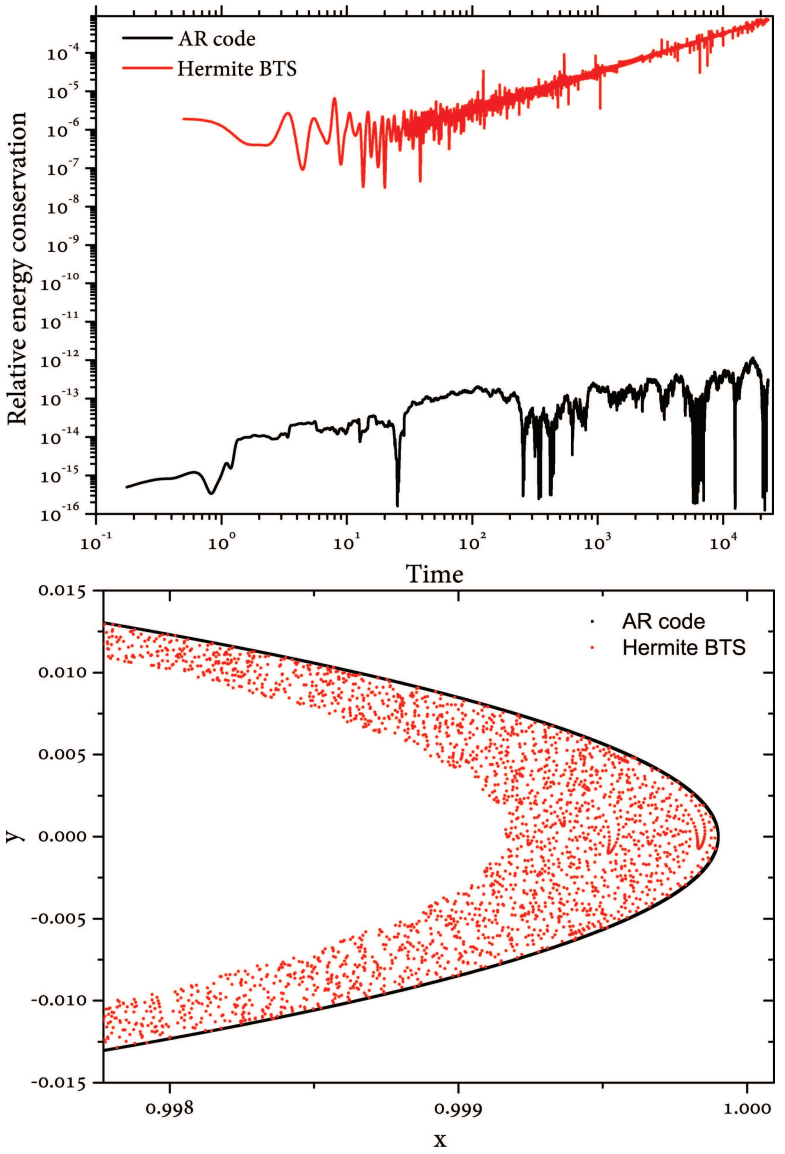


**Figure 5.4:** Case of eccentricity $e = 0.96$. Relative errors in total energy (top panel) for a standard (red) an regularized (black) algorithm. The bottom panel shows the positions (around apocentre) obtained by the standard (red) and regularized (black) integrators evolving the system over a time interval corresponding to $\sim 10^4$ orbital revolutions of the lighter star around the more massive particle.

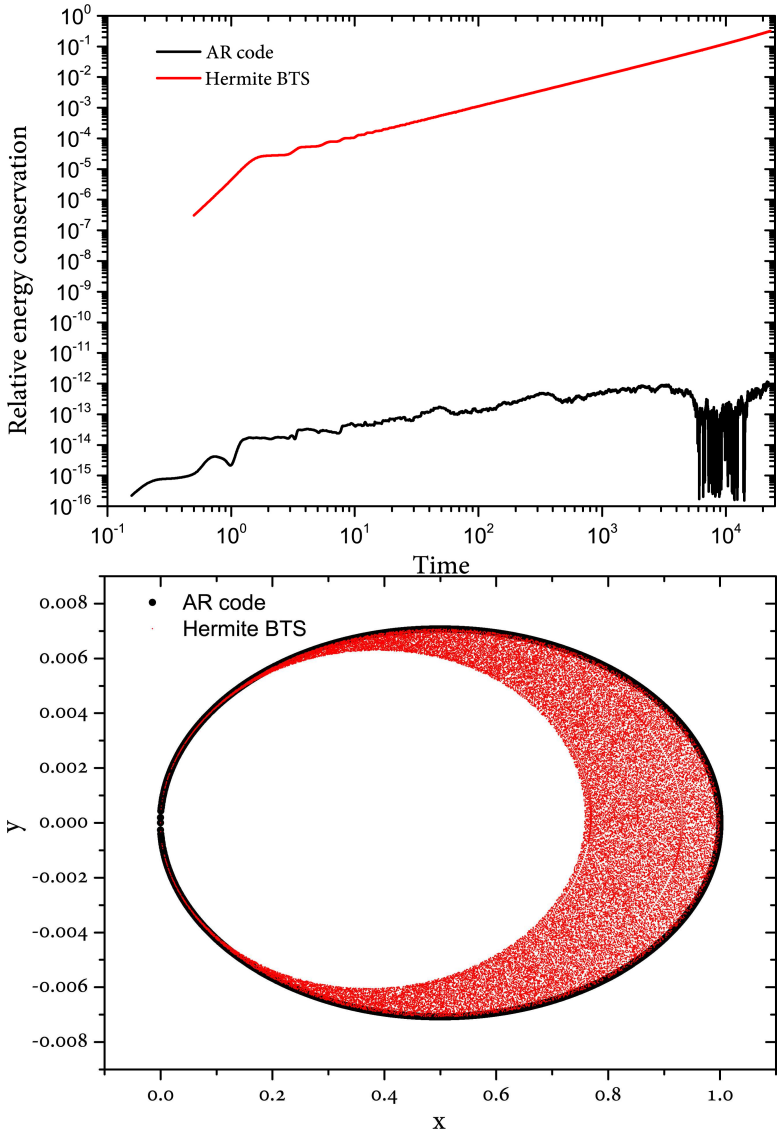


**Figure 5.5:** Case of eccentricity $e = 0.9999$. Relative errors in total energy (top panel) for a standard (red) an regularized (black) algorithm. The bottom panel shows the positions (around apocentre) obtained by the standard (red) and regularized (black) integrators evolving the system over a time interval corresponding to $\sim 10^4$ orbital revolutions of the lighter star around the more massive particle.

Figures 5.2, 5.3, 5.4 and 5.5 show a detail of the trajectory (sampled accordingly to the variable time steps of the integrators) at apocentre for different eccentricities flanked by a plot which shows the relative energy variation in function of time using the regularized code (AR) and using a standard Hermite's algorithm with BTS. The accuracy parameter for the Hermite's code has been chosen equal to $0.002$ and the tolerance of the BS integrator in our implementation of the AR (see [84] for details) equal to $10^{-14}$ in order to get very high accuracy in both cases (typical values of these parameters are respectively $0.01$ and $10^{-12} \div 10^{-13}$). The period $P$ of the orbit in terms of the parameter $\zeta$ can be written as

$$P = \frac{2\pi}{\sqrt{G\,(M+m)}} \left( \frac{R_a}{2-\zeta^2} \right)^{\frac{3}{2}}. \tag{5.54}$$

Each systems has been evolved for $10^4$ periods which means that choosing $G = M = 1$, $m = 0.0001$, $R_a = 1$ and $\zeta \in (0;1)$ we have $P \in (2.2;6.3)$. As we can see in figures 5.2, 5.3, 5.4 and 5.5, the relative energy conservation reached by the AR code is constantly $\sim 8$ orders of magnitude better than that reached by the standard Hermite's integrator. Obviously, the difference is more pronounced for the extreme eccentric orbit represented in Fig. 5.5. The worse energy conservation reflects in the accuracy in determining the positions (and velocities) of the two bodies along their orbit. In fact, in the right panels of figures 5.2, 5.3, 5.4 and 5.5 we can see how the cumulation of the error in the Hermite's scheme causes the orbit to shrink, that is the apocentre distance is not conserved and the orbit does not form a perfectly closed loop. On the other hand, the points (black) obtained using our implementation of the AR regularization seem to draw, by eye, a perfect ellipse in all the shown cases. Tab. **??** summarizes the obtained results including also the times needed to complete the integrations in seconds. It is possible to see that the AR code is constantly more expensive, in terms of computing time, than the Hermite's integrator. This is true provided that the accuracy parameters are $0.002$ and $10^{-14}$ respectively. In fact, we can see in Tab. **??** that increasing the accuracy parameters of one or-

der of magnitude the AR code is both significantly more accurate and faster executing all the tested cases, paying $\sim 2$ orders of magnitude of worse relative energy conservation although its value ($\sim 10^{-10}$) remains very good. Nevertheless, for $N \gtrsim 10$, using the AR code becomes not convenient any more (despite its very high accuracy) because of its complexity mainly due to chain transformations, chain inspections, BS integrator, frequent evaluation of the gravitational potential energy, etc ....

Accuracy parameters : $\eta_{hermite} = 0.001$, $\eta_{BS} = 10^{-14}$

| $\zeta$ | $e$ | $t_H\,(s)$ | $t_{AR}\,(s)$ | $\left(\frac{\Delta E_{max}}{E}\right)_H$ | $\left(\frac{\Delta E_{max}}{E}\right)_{AR}$ |
|---|---|---|---|---|---|
| 1 | 0 | 2 | 35 | $\sim 10^{-5}$ | $\sim 10^{-13}$ |
| 0.5 | 0.75 | 3 | 35 | $\sim 10^{-3}$ | $\sim 10^{-12}$ |
| 0.2 | 0.96 | 4 | 35 | $\sim 10^{-2}$ | $\sim 10^{-12}$ |
| 0.01 | 0.9999 | 25 | 35 | $\sim 10^{-1}$ | $\sim 10^{-12}$ |

**Table 5.1:** This table summarizes the results obtained in our tests. $t_H$ and $t_{AR}$ are respectively the times needed to complete the preformed simulations, in seconds, given the accuracy parameters written at the top of the table.

Accuracy parameters : $\eta_{hermite} = 0.01$, $\eta_{BS} = 10^{-13}$

| $\zeta$ | $e$ | $t_H\,(s)$ | $t_{AR}\,(s)$ | $\left(\frac{\Delta E_{max}}{E}\right)_H$ | $\left(\frac{\Delta E_{max}}{E}\right)_{AR}$ |
|---|---|---|---|---|---|
| 1 | 0 | 2 | 2 | $\sim 10^{-4}$ | $\sim 10^{-12}$ |
| 0.5 | 0.75 | 2 | 3 | $\sim 10^{-2}$ | $\sim 10^{-11}$ |
| 0.2 | 0.96 | 3 | 3 | $\sim 10^{-1}$ | $\sim 10^{-10}$ |
| 0.01 | 0.9999 | 20 | 3 | $\sim 1$ | $\sim 10^{-10}$ |

**Table 5.2:** This table summarizes the results obtained in our tests. $t_H$ and $t_{AR}$ are respectively the times needed to complete the preformed simulations, in seconds, given the accuracy parameters written at the top of the table.

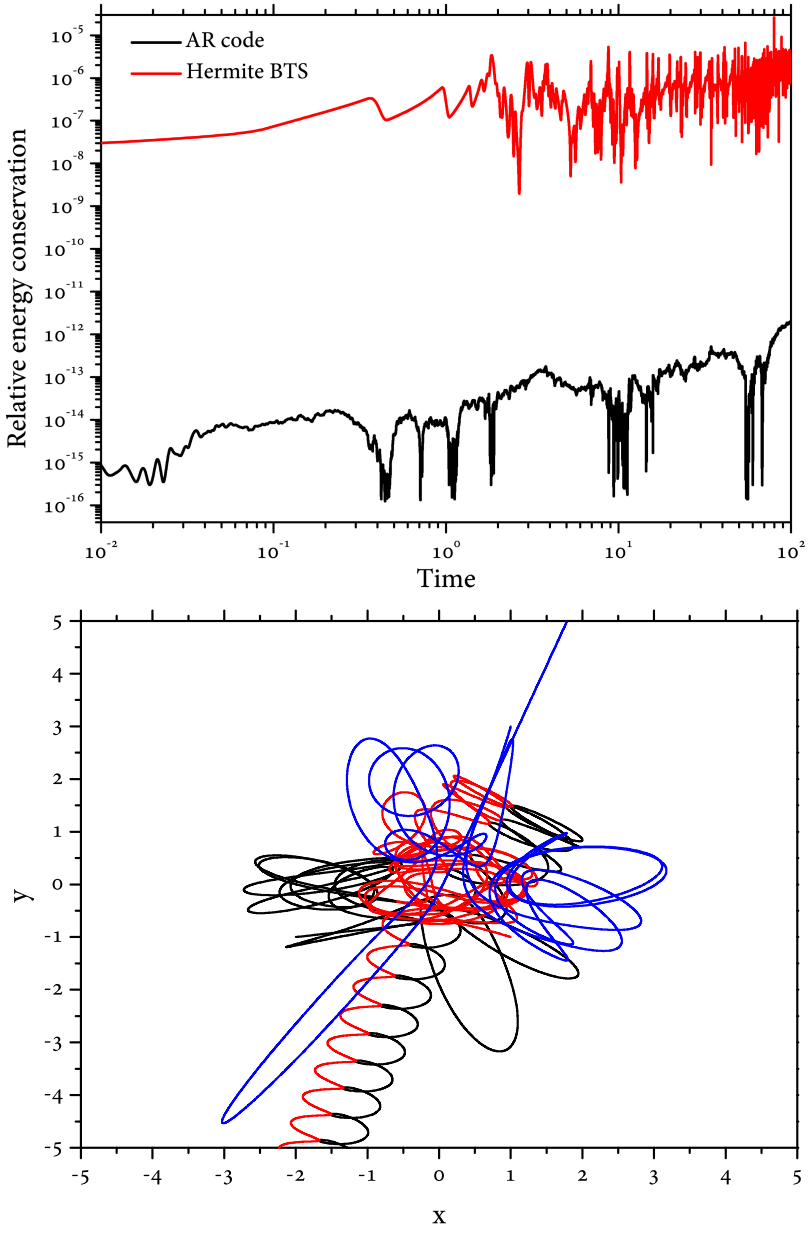


**Figure 5.6:** Results in relative energy variation (top panel) and trajectories (bottom panel) for the Pythagorean 3-body problem as integrated by the two tested codes. The trajectory is shown only for the AR code because, as explained in the text, the Hermite's method could not reproduce the exact result.

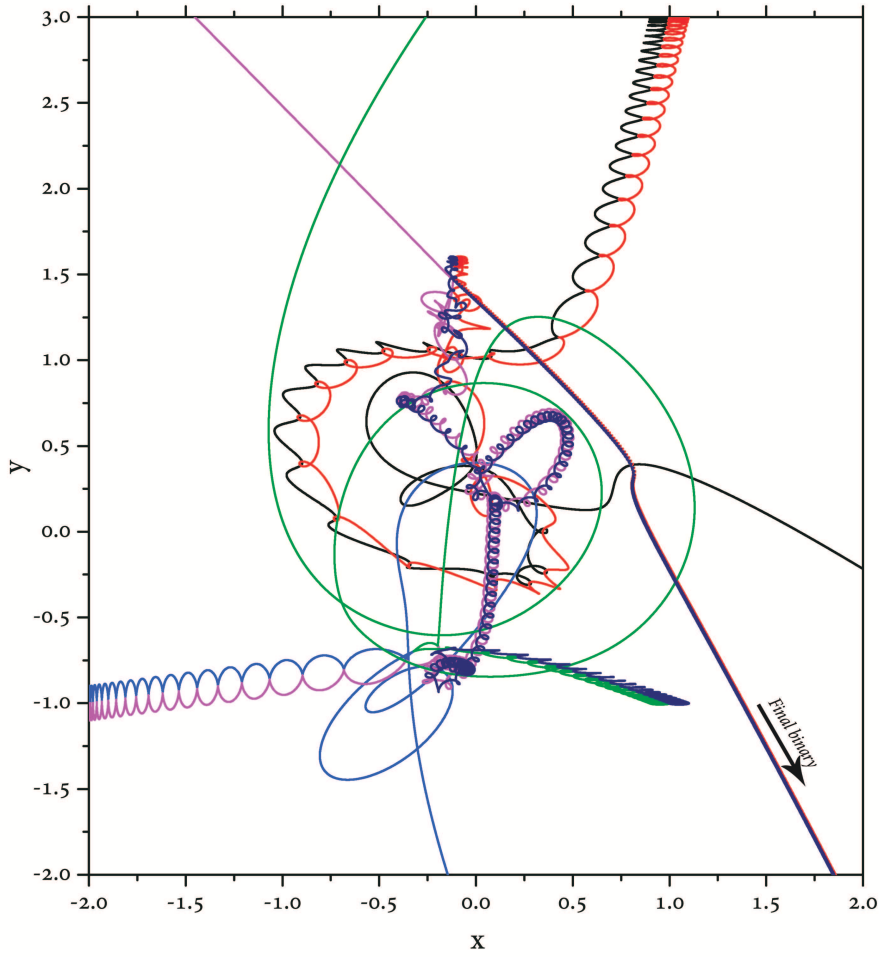


**Figure 5.7:** The trajecotries obtained using our implementation of the Mikkola's algorithmic regularization, integrating the so called Pythagorean 6-body problem. Three binaries start at the vertices of a right triangle with null velocities. The evolution shows several exchanges of the components of the binaries and the final result is the disruption of the system and the formation of a very tight binary formed by the stars colored respectively in red and blue.

In Fig. 5.6 and 5.7 we show the last two tests for our AR code. Fig. 5.6 represents results obtained integrating the so called *Pythagorean 3-body problem* which consists in 3 bodies of (numerical) mass 3, 4 and 5 placed with null velocities at the vertices of a right triangle of sides

of length 3, 4 and 5. This system has been shown to be very chaotic (see for example the paper by Aarseth et al. [5]) but it has been shown that the final net result is the formation of a binary and the ejection of the lightest mass, thing that occurs at $t \simeq 60$ [90]. This represents a very hard test for a $N$-Body code because even very small errors in the integration of close encounters produce very big difference of the final configuration. In fact, in Fig. 5.6 we show the results of the integration between $t = 0$ and $t = 80$ using only the AR code because the Hermite's integrator, independently from the value of the accuracy parameter, cannot reproduce the final correct result which is, on the contrary, shown in the right panel of Fig, 5.6 for the regularized code. The left panel shows the relative energy variation. For the Hermite's code we showed the curve relative to $\eta = 0.001$ which gave the best result. The total computing time was $\sim 0.1$ seconds for the AR code while it gets to $\sim 0.5$ seconds using the Hermite's scheme. Fig. 5.7 represents the Pythagorean 3 body problem where the three bodies are replaced by 3 binaries (therefore a 6-Body problem). The situation is even more critical with respect to the standard Pythagorean case; in fact the Hermite's integrator immediately stops because the integration time step becomes prohibitively small. The only chance in this case is to use a regularized code which is also very fast ($\sim 3$ seconds to integrate $20$ time units).

# 6 The emerging state of open clusters after their violent relaxation

## 6.1 Introduction

This chapter presents some preliminary results, on which we are currently working, about the segregation of masses which occurs on very short time scales (significantly smaller than the relaxation time) as a direct consequence of violent dynamical processes. This constitutes one of the possible astrophysical applications of the instruments already discussed in the previous chapters of this work.

Astronomical observations show that several stellar systems (from young and very young open star clusters to rich clusters of galaxies) manifest a certain degree of segregation of the most luminous and massive components in their inner regions. As an example, the Orion Nebula Cluster (ONC) has been found to be mass segregated down to about $5M_{\odot}$ (see for example [52]) despite its young age which has been estimated to be less than 2 Myr. The main cause of the rapid mass segregation process for such systems is still under debate. In particular, a dynamical origin is usually excluded because the estimation of the age of some stellar systems is less than their two-bodies relaxation time which is considered the time-scale needed to segregate masses. Specifically, the time needed

by a system to get to a dynamically relaxed state, as we have already discussed in section 1.2.4, is defined as

$$t_{rel} \equiv \frac{v^2}{D\left[\left(\Delta v_{\parallel}^2\right)\right]} \tag{6.1}$$

where $v$ is the typical velocity of a star in the system, and $D\left[\left(\Delta v_{\parallel}^2\right)\right]$ in one of the three independent diffusion coefficients which derive from the *Fokker-Planck approximation* for the master equation (see [18]). For simplicity, if we assume that the velocity distribution of the field stars is Maxwellian with dispersion $\sigma$, it is possible to obtain an explicit expression for $D\left[\left(\Delta v_{\parallel}^2\right)\right]$ and the equation 6.1 can be rewritten as

$$t_{rel} = \frac{v^2 \sigma X}{4\sqrt{2}\pi G^2 \tilde{\rho} \tilde{m} \ln \Lambda G(X)} \tag{6.2}$$

where $\tilde{\rho}$ is the mean mass density of the field stars, $\tilde{m}$ the mean stellar mass, $\ln \Lambda$ is the Coulomb logarithm, $X \equiv \frac{v}{\sqrt{2}\sigma}$ and $G(X)$ is a function that can be expressed as

$$G(X) = \frac{1}{2X^2}\left[\mathrm{erf}(X) - \frac{2X}{\sqrt{\pi}e^{-X^2}}\right]. \tag{6.3}$$

Equation (6.2) is valid whether the initial conditions are, indeed, not too far from virial equilibrium which is not necessary a correct assumption especially if we consider the early dynamical evolution of young stellar systems whose stars form in regions which are observed to be sub-structured, clumpy and in sub-virial conditions. Farouki et al. [42] and Allison et al. [9] have already shown, through numerical $N$-Body simulations, that, if the initial state of a stellar system is not in equilibrium and the initial distribution of the positions of the stars is not

homogeneous, a significant degree of mass segregation might be observed on very short time-scales which are significantly shorter than the two-bodies relaxation time of the considered system. In particular, they argue that the main mechanism which brings to mass segregation on very short time scales resides in the short interval of time in which the initially violent collapse creates a dense core containing about half the mass of the stellar system in a radius of about one tenth of its characteristic dimension in the initial state. They showed that the time to segregate masses down to $4 - 5M_{\odot}$ in the dense core is comparable to its living time (approximatively $0.1$ Myr). This is enough to justify also the degree of mass segregation observed in some astrophysical systems (like the above cited ONC). On the other hand, some works, like that by Bonnell and Davies [20], excluded that the observed mass segregation in young stellar clusters could be due to a violent dynamical evolution introducing, rather, the hypothesis of an in situ formation of the most massive and brilliant stars. Bonnell and Davies [20], in their work, investigated the dynamical evolution of both spherical stellar systems initially in virial equilibrium and not-spherical stellar systems in sub-virial conditions. They found that the time-scale for mass segregation was largely unaffected by differences in initial phase-space distribution of the stars. It is evident that, still nowadays, we cannot discriminate between the two formulated interpretations to explain the phenomenon of rapid mass segregation even because the astronomical observations of star forming regions are very difficult to accomplish because it is often needed to see through very dense gas clouds. Therefore it is very difficult to prove that the most massive stars form primordially very close to the innermost regions or if the system segregates masses later. Another dynamical mechanism, which has not been deeply investigated yet (even if already highlighted by Aarseth et al. [6] and McMillan et al. [70]), which might play an important role in segregating masses, is the initial rapid fragmentation of such stellar systems whose stars are initially homogeneously distributed in spherical symmetry with approximatively null velocities. In particular, McMillan et al. [67] stressed that

the sub-systems which form during the collapse phase show, just before the bounce of the system, a certain degree of mass segregation which is preserved after the collapse, excluding the hypothesis that the system segregates masses during the formation of the short-living core.

There is not a clear scenario which is even more complicated if the presence of a certain number of primordial binaries or that of a background gas (both not considered in $N$-Body simulations so far) are included. In particular, in this work we study the effects of violent collapse of an $N$-body self-gravitating system starting from initially cold conditions (initial virial ratio $Q = 0.0$) and an homogeneous distribution of stars. We also check the role played on mass segregation, on the resulting density profile and on the stars velocity dispersion taking into account the presence of both residual gas after star formation and of a stellar mass black hole. For all our $N$-Body simulations we consider the simplest case of a bimodal mass spectrum with bodies initially distributed randomly in a sphere with initial radius $R = 1$. We report here some preliminary results of the simulations performed. Firstly we describe the models of stellar systems adopted to give, after, a description of both the software and hardware resources used to dynamically evolve them. Finally we present and discuss the preliminary results with attention to both the physics of violently relaxing intermediate $N$-body systems and to possible comparisons with observational data of real clusters.

## 6.2 Modelization

We performed a large set of direct $N$-Body simulations of young star clusters composed by a number of bodies between $N = 128$ and $N = 1024$. For each simulation we used a bimodal mass spectrum dividing

stars into light (each with mass $m_L$ and total number $N_L$) and heavy (each with mass $m_H$ and total number $N_H$) such that

$$\frac{m_H}{m_L} = P \qquad \frac{N_H}{N_L} = Z. \tag{6.4}$$

In this paper we investigate the simplest models considering only $P = 2$ and $Z = 1$. Initially the bodies have been distributed randomly in a sphere with radius $R = 1$ varying the initial virial ratio $Q$ of the system (see chapter 1). We ran simulations of different stellar systems using values of $Q$ from $Q = 0$, which corresponds to the most violent collapse, to $Q = 1$, which means dynamical equilibrium, using a variation step of $0.1$. These values have been chosen taking into account that observations of young stellar systems suggest that stars form in clusters in sub-virial equilibrium (see, for example, [63] and [57]) so it is reasonable to choose always an initial virial ratio $Q < 1$. We also investigated the presence of gas which is modelled as an analytical additional contribution to the accelerations of stars. We represented this external field using a Plummer model, therefore the gas is described by the potential

$$\Phi_P(r;t) = -\frac{GM_G}{\sqrt{r^2 + r_c^2(t)}} \tag{6.5}$$

where $r_c(t)$ is the gas core radius and $M_G$ is the gas total mass. We modelled our models such that the gas core radius may vary in time according to the formula

$$r_c(t) = r_0 e^{\frac{t}{\tau}} \tag{6.6}$$

where $r_0 = r_c(t = 0)$ and $\tau$ is a characteristic time scale for the gas core radius variations. By including formula 6.6 in our $N$-body simulations, we can study the dynamical evolution of stellar systems considering also an expanding gas which mimics the removal of molecular clouds, mainly due to stellar winds, in which such young clusters are still embedded. We also varied the parameter $\tau$ to model different expansion rates using $\tau = +\infty$ to represent a stationary gas and $\tau = 1$ to emulate a process of gas removal which act on a time scale compara-

ble with the system crossing time. In our simulations we included also a particle with a mass significantly higher that the other bodies. This point-mass may be considered as a stellar mass black hole whose mass will be indicated, hereafter, as $m_{BH}$. We compared the resulting mass segregation, density profile and velocity dispersion when the black hole is included and when it is not. The black hole mass is such that

$$m_{BH} = K m_L \tag{6.7}$$

where $K$ is an integer value which, in our work, assumes the values $K = 0$ and $K = 50$. In any case the total mass of the stellar system $M_C$ is thought to be constant and unitary, i.e.

$$M_C = N_H \cdot m_H + N_L \cdot m_L + M_G + m_{BH} = M_* + M_G = 1 \tag{6.8}$$

where we used the notation $M_*$ to indicate the total mass in stars. The total number of particles is

$$N = N_H + N_L + N_{BH} \tag{6.9}$$

where $N_{BH} = 1$ only if a black hole is present, otherwise $N_{BH} = 0$. In order to determine the fraction of gas in which the stellar system is embedded we used the star formation efficiency (hereafter $\mathcal{S}$) parameter which is

$$\mathcal{S} = \frac{M_*}{M_G + M_*} = \frac{M_*}{M_C}. \tag{6.10}$$

If we fix the value of $\mathcal{S}$, we can determine the stellar mass of the cluster $M_* = \mathcal{S} M_C$ and, as a consequence, we get $M_G = M_C - M_* = M_C(1-\mathcal{S})$. We considered simulations with $\mathcal{S} = 1$ (no gas included) and $\mathcal{S} = 0.3$ which is a likely astrophysical value. Because we are interested in the emerging state of young and very young open clusters we evolved our stellar systems shortly in time ($< 8\mathrm{Myr}$) therefore we neglected the effects of stellar evolution. For each set of parameters we performed 25 runs to have a certain degree of statistics to present our final results.

To perform direct $N$-body simulations we used our highly parallel $N$-body code `HiGPUs` running on our private machine containing a CPU Intel i7 950 and 2 nVIDIA Tesla C2050 (Fermi) cards. Although the GPU used is not the best in terms of cost and computing capability, being, nowadays, and old generation card, thanks to higher core frequencies with respect to the most modern GPUs, it performs well in regimes of weak load (that is using a number of particles less than about 1024, as we described in 3.5).

## 6.3 Results

To measure the degree of mass segregation we used, first of all, the ratio between several lagrangian radii of heavy and light particles so that a value significantly greater than 1 indicates the presence of mass segregation. Lagrangian radii are very good indicators of mass segregation until the system to study has spherical symmetry but, in our simulations, this is true only after the first bounce which, in our units, occurs at time $t = t_B \simeq 1$. In fact, for $t \lesssim t_B$ we observe a fragmentation of the system resulting in the formation of various clumps which are not spherical. Therefore, we used also the so called *Minimum Spanning Tree* method[1] developed by Allison et al. [10]. Given a sub-set of points indicated with the letter $K_m$, composed by $m$ points, belonging to a system composed by $m^{'} > m$ bodies, the degree of mass segregation established for that sample, $\Lambda_{K_m}$, is defined as

$$\Lambda_{K_m} = \frac{\langle l_{norm} \rangle}{l_{K_m}} \pm \frac{\sigma_{norm}}{l_{K_m}} \tag{6.11}$$

where $l_{K_m}$ is the MST for the sample $K_m$, $\langle l_{norm} \rangle$ is the average MST for $m$ randomly selected stars in the whole system and $\sigma_{norm}$ is its associated standard deviation. To calculate $\langle l_{norm} \rangle$ we averaged the results

[1] The Minimum Spanning Tree (MST) is the shortest path length which connects a certain number of points without forming close loops.

got, for each run, from 200 different sub-sets of points. From the definition given in 6.11 it is evident that the generic sample $K$ is mass segregated if its corresponding value of $\Lambda_K$ is significantly grater than 1. When using the MST method to investigate the distribution of masses in our simulations the process of removing escapers from the numerical results was needed. This is important because, after the first bounce, a significant amount of mass (almost equally divided between heavy and light particles and approximatively quantifiable as 20% of the total mass) is lost; therefore a single star, far from the stellar system core, may alter significantly the length of the spanning tree of a specific population. To identify escapers correctly one of the best methods is to adopt criteria on both energy (which should be positive) and distance from the most dense region of the system (the core). For our purposes, it was enough to use a truncating distance of $d = 1$.

### 6.3.1 1024 stars, no gas, no central black hole

This is the simplest case we studied. We generated 25 different samples changing the seed of the *Mersenne Twister* random number generator [66] and here we show the results obtained from these simulations.

Fig. 6.1 shows the ratio between the averaged value of $\Lambda$ for the heavy ($\Lambda_H$) and for the light ($\Lambda_L$) stars in function of time where the time unit is the initial system crossing time. Several curves, which correspond to different values of the initial virial ratio, are represented. The first important evidence is that the observed degree of mass segregation depends strictly on the initial state of the system; the farther from equilibrium the higher and the quicker the degree of the resulting mass segregation on both short and long time scales. Another important result which is worth noting is that a significant degree of mass segregation is obtained starting from homogeneous and smooth initial conditions. This implies that starting form an initially clumpy distribution is not

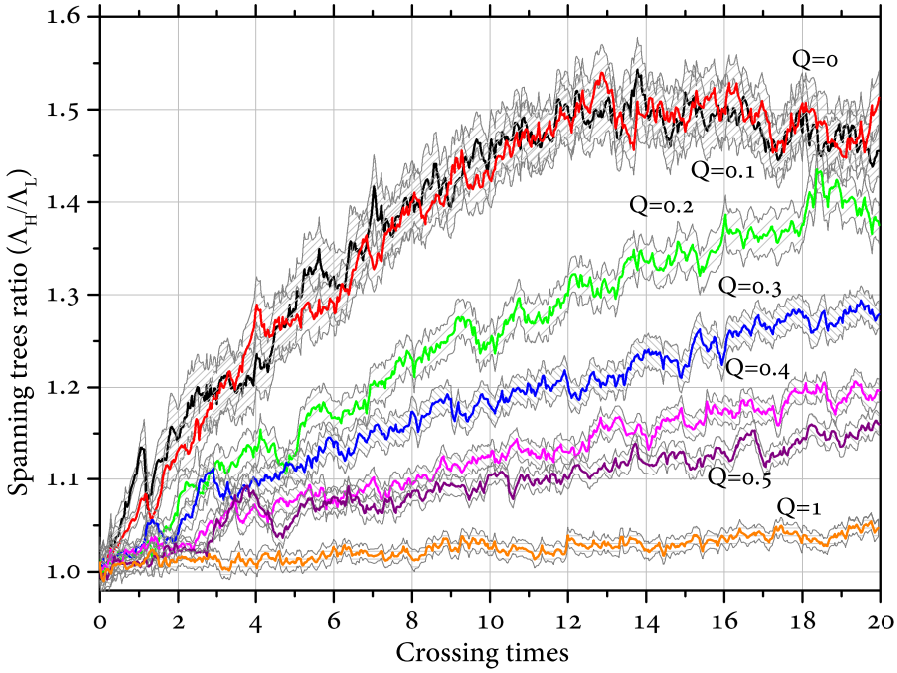


**Figure 6.1:** It shows, as a function of time expressed in units of the initial system crossing time, the ratio between $\Lambda_H$ and $\Lambda_L$ for several values of the initial virial ratio $Q$. Values of $Q$ between $0.6$ and $0.9$, both included, are not showed in order to obtain a more clear representation of the results. Each curve represents an average value of $\Lambda_H/\Lambda_L$, taking into account the results obtained from the single runs, while the error (standard deviation) is represented by the pattern area.

needed. For the cases of very violent collapses ($Q = 0.0$ and $Q = 0.1$) the system gets to a saturation of the degree of mass segregation around 12 time units while for other values of the initial virial ratio mass segregation continues at approximatively constant rates.

Fig. 6.2 shows a detail of the Fig. 6.1. This is a zoom which spans an interval of time between $t = 0$ and $t = 2$ that is from the initial state to just after the first bounce. The graphical evolution of the cluster in this interval of time has been summarized with 4 snapshots collected in Fig. 6.3. As we can see in Fig. 6.2, there is a rapid increase of mass

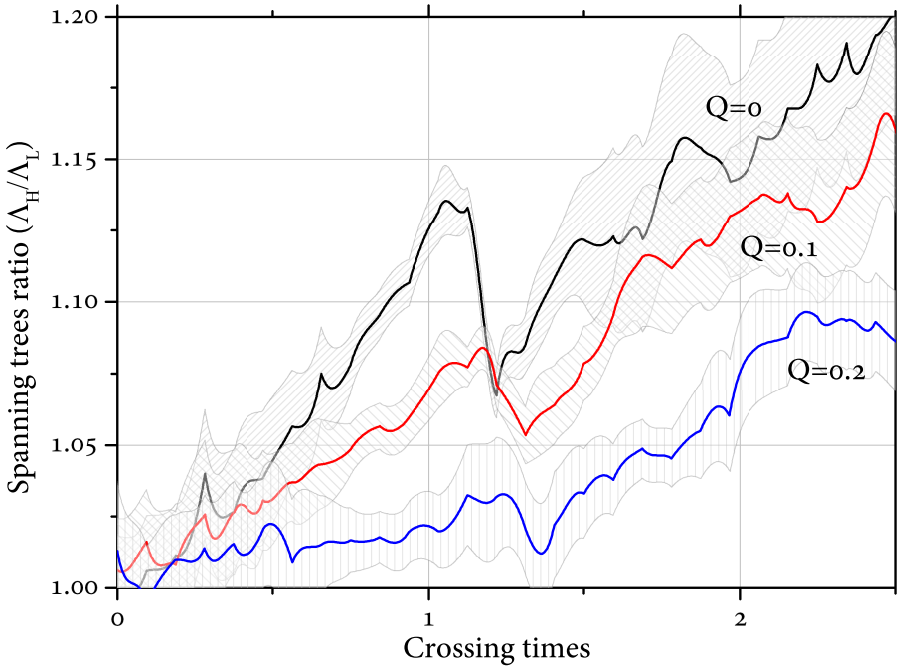


**Figure 6.2:** It shows a detail of 6.1 zooming in the interval of time between 0 and $\sim 2$. The curves are for $Q = 0$, $Q = 0.1$ and $Q = 0.2$.

segregation from $t = 0$ to $t \simeq 1$ then an inverse trend is observed approximatively around the time $t \simeq t_B$ which corresponds to the state of maximum compression (see Fig. 6.3) and, finally, mass segregation starts again with the same rate and with the same efficiency as it was before the bounce. The rapid increase of mass segregation before the bounce is in perfect agreement with the mass segregation observed in the sub-clumps that form as the system collapses. Nevertheless, when the merging process begins, the degree of mass segregation does not seem to be preserved and Fig. 6.2 shows an inverse trend. After the merging process, mass segregation continues with the same efficiency as it was before the bounce but, this time, sub-structures have been completely removed, therefore, mass segregation continues inside the dense core which forms just after the bounce. It is clear that sub-clumps cannot be considered the only cause of the observed mass segregation

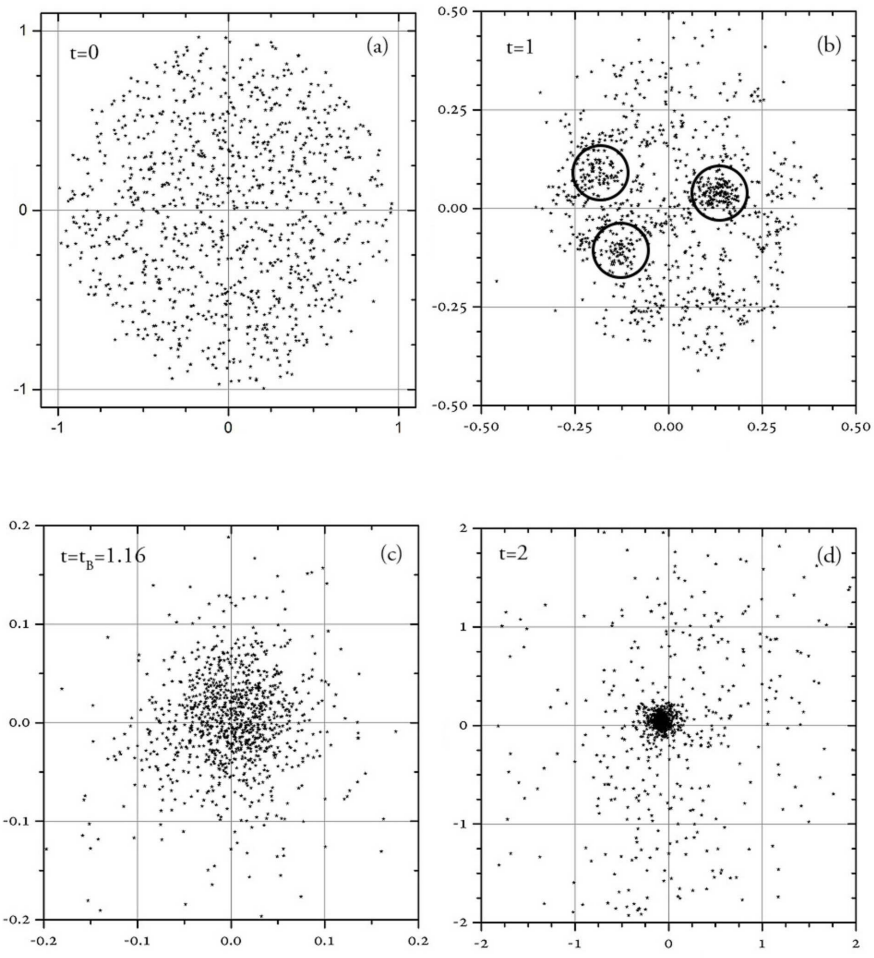


**Figure 6.3:** This is a visual representation of one of the simulated clusters at four different times. Panel *a* shows the homogeneous initial distribution, panel *b* the formation of several sub-clumps of which the most evident have been circled, panel *c* represents the state of maximum compression of the stellar system while panel *d* is a view of the cluster after the bounce.

on short time scales, and the same can be said for the short-living but very dense core. Rather, the two phenomena have to be taken into account simultaneously because the first one acts on time-scales such that $t \lesssim 1$ while the second one is responsible of the long lived mass segregation for $t \gtrsim t_B$. Times around the collapse phase ($t \simeq t_B$) corresponds

to a transition between the two regimes. Nevertheless, this interpretation is valid provided that $Q \lesssim 0.3$; in fact, for $Q \gtrsim 0.3$ sub-clumps do not form at all although a degree of mass segregation significantly greater than the equilibrium case ($Q \simeq 1.0$) is established. In these cases ($Q \gtrsim 0.3$) the only mechanism responsible for the rapid and secular mass segregation has to be the dynamical evolution of the dense core which forms after the collapse. For completeness, we also show

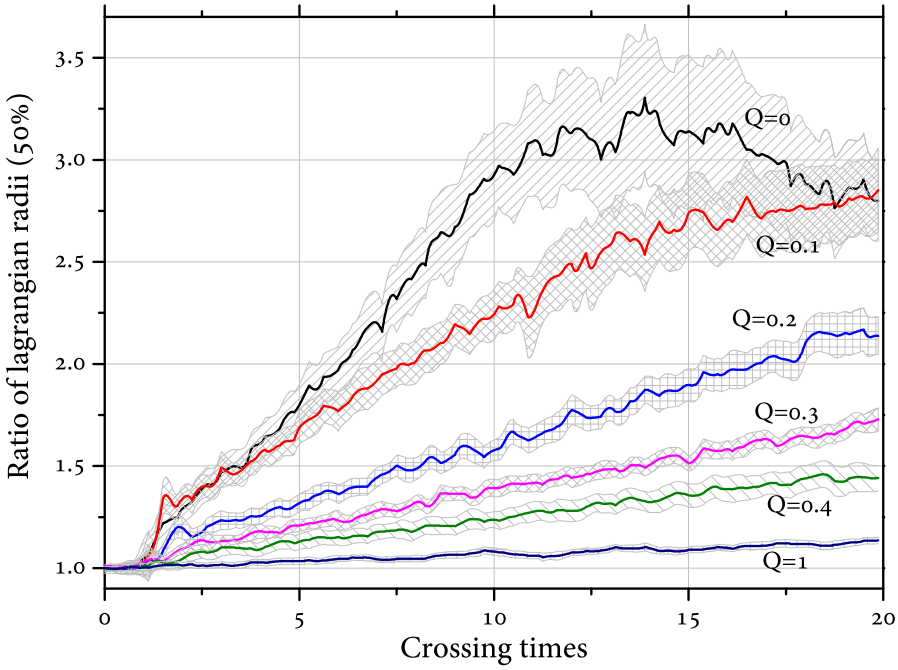


**Figure 6.4:** Ratio between the lagrangian radius, containing 50% of the total mass of a specific sample of stars, of light and heavy particles. We do not show data related to other lagrangian radii for clarity because the resulting curves are almost indistinguishable from those reported here. Different curves correspond to different values of the initial virial ratio of the system.

in Fig. 6.4 the results obtained studying another indicator of mass segregation that is the ratio between various lagrangian radii of light and heavy stars. The ratio of lagrangian radii can be used to quantify the degree and the efficiency of the mass segregation when the studied system maintain spherical symmetry which is true, in our case, provided

that the interval of time studied is grater than $\sim t_B$. In other words, we use lagrangian radii to point out the presence of a long lived mass segregation whose efficiency depends on the above described phenomena which occur at $t \lesssim t_B$. In Fig. 6.4 we show the ratio calculated using the lagrangian radius containing 50% of the total mass of, respectively, light and heavy stars starting from various values of the initial virial ratio. The phenomenon of mass segregation, for $Q = 0$, is evident in fact, on average, in 6 crossing time, the typical dimension of the spatial region occupied by light stars is almost twice that of the heavy particles. This is very efficient if compared with the case of $Q = 1$ whose curve is almost flat. This confirms again the strong dependence of the efficiency of mass segregation on the initial conditions of the system.

### 6.3.2 1024 stars, no gas, central black hole included

As we can see in Fig. 6.5 the presence of a central massive particle, whose mass is 50 times that of a generic "heavy" star, tends to compact the curves of the spanning trees ratios with respect to the case shown in Fig. 6.1. In fact, the process of mass segregation is less efficient (the values of the spanning trees ratio are less than that reached in Fig. 6.1) but still evident and strongly dependent on the violence of the collapse. In particular, the presence of mass segregated structures before the bounce is still present at least for $Q = 0.0$ but clumps disappear already for $Q \gtrsim 0.1$, that is significantly before with respect to what argued analysing the results deriving from the simulations of a system which does not contain a central heavier star, for which substructures did not form for $Q \gtrsim 0.4$. These differences can be due to the reduced efficiency of close encounters (that is, redistribution of kinetic energy) between stars due to the presence of a strong contribution in the gravitational potential due to the central black hole. The immediate consequence of having less efficient exchanges of energy is that mass

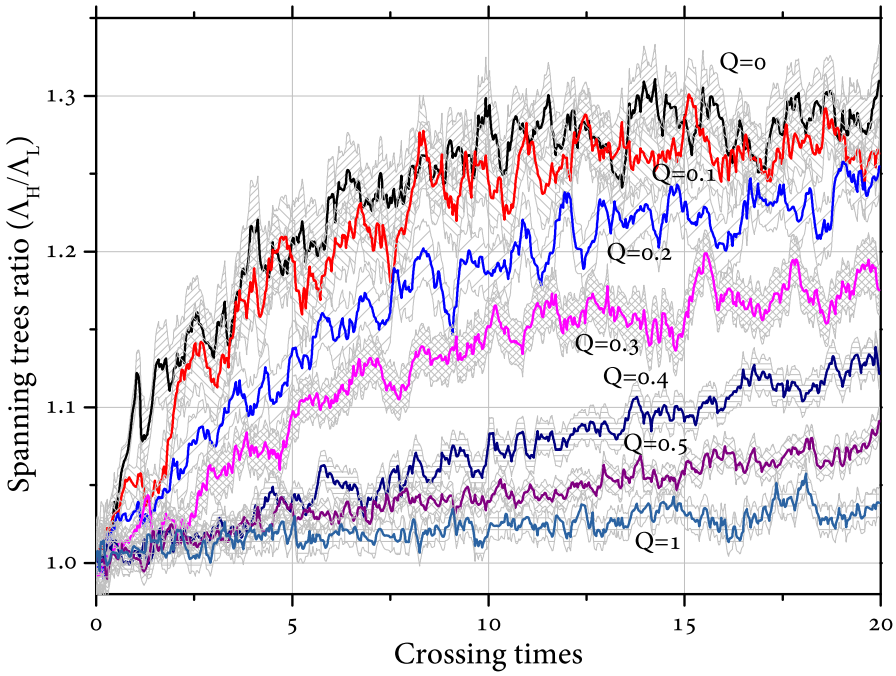


**Figure 6.5:** It shows, as a function of time expressed in units of the initial system crossing time, the ratio between $\Lambda_H$ and $\Lambda_L$ for several values of the initial virial ratio $Q$. Values of $Q$ between $0.6$ and $0.9$, both included, are not showed in order to obtain a more clear representation of the results. Each curve represents an average value of $\Lambda_H/\Lambda_L$, taking into account the results obtained from the single runs, while the error (standard deviation) is represented by the pattern area.

segregation is reduced, with respect to the case described in section 6.3.1, even if it is still present.

### 6.3.3 1024 stars, gas included, no central black hole

It is known that the majority of young and very young star clusters are still embedded in their native proto-cloud. This residual gas can be important in terms of background potential affecting significantly the

dynamical evolution of the stellar cluster. In this work we modelled the presence of a gas using a simple model of a stationary background added as a further analytical contribution to the mutual gravitational interaction between the bodies in the $N$-Body system. This may not represent strictly the astrophysical reality because the gas distribution is, generally, unknown with enough accuracy but, using our simple model, we can study the overall effect of the inclusion of a background potential. We verified that the analytic gravitational potential smooths both the cluster potential and the 2-body close encounters, decreasing the efficiency of mass segregation on both small and longer time scales.

Figure 6.6 is the analogous of Fig. 6.1 when a stationary background is included. As we can see, the results got for the case of initial virial ratio $Q = 0$ are very similar to that obtained for the case in which no gas was included; it is still evident the formation and quick mass segregation of sub-clumps, before the bounce, as much the inverse trend of the spanning trees ratio around the time corresponding to the state of maximum compression of the system. The mass segregation on longer time scale is slightly less efficient than that observed for the case $Q = 0$ in Fig. 6.1. For values of the initial virial ratio such that $Q \gtrsim 0.1$ the situation deeply changes. In fact, the presence of gas smooths close encounters of stars and mass segregation is not efficient neither on small and long time scales. We are currently trying to investigate deeper this point but it is interesting to note that, for the case $Q = 0.1$, profound differences in mass segregation between the case which includes gas and that in which the gas is not present are seen also on the evolution of the distribution of the velocities of stars. In the case $Q = 0.0$ no significant differences emerge.

In Fig. 6.7 we show the distribution of the velocities of the stars at different times. Initially, at $t = 0$, the velocity distribution follow more or less the same trend. The situation deeply changes after 1 crossing time: the velocity dispersion in the system which does not include gas increases

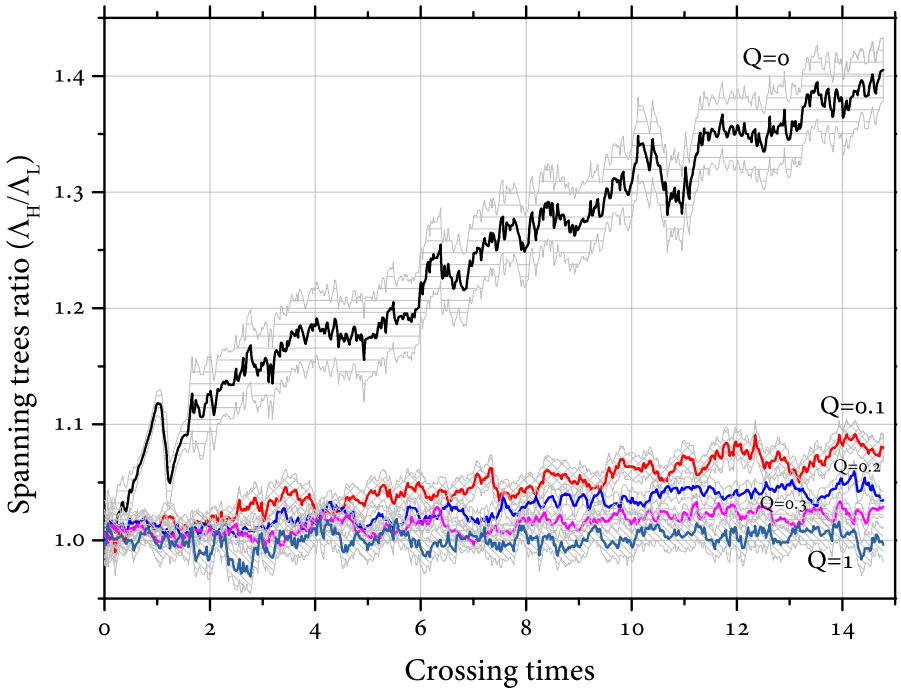


**Figure 6.6:** It shows, as a function of time expressed in units of the initial system crossing time, the ratio between $\Lambda_H$ and $\Lambda_L$ for several values of the initial virial ratio $Q$ when a component of stationary background, which mimics the presence of a gas, is added to the newtonian interaction force. Values of $Q$ between $0.4$ and $0.9$, both included, are not showed in order to obtain a more clear representation of the results. Each curve represents an average value of $\Lambda_H \Lambda_L$, taking into account the results obtained from the single runs, while the error (standard deviation) is represented by the pattern area.

more rapidly than that of the system which includes the gaseous contribution. This means that the background potential reduces the efficiency of the 2-body interactions and than the possibility to form gravity-driven substructures (clumps) and, at the same time, the efficiency of mass segregation. At the bounce, the difference between the two considered systems is even more pronounced and, after 2 crossing times, the two distributions are completely different in shape. In particular, the presence of the gas tends to crush the distribution on the y-axis that is particles

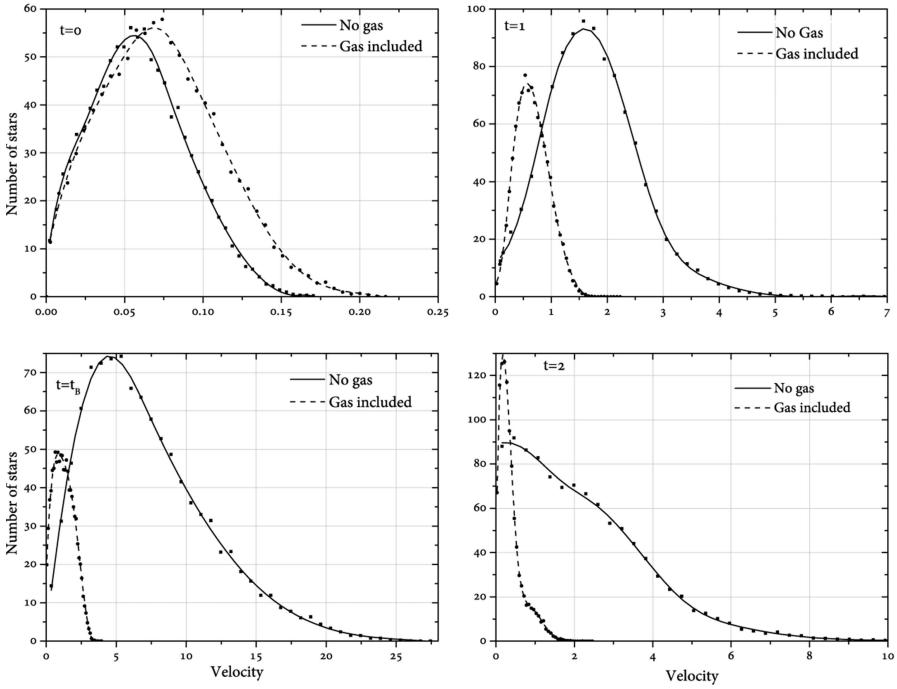


**Figure 6.7:** The figure shows the distribution of the velocities of stars, in the case of $Q = 0.1$, considering the presence of a background (stationary) gas (dashed line) and a system which does not contain any background potential (solid line). The four panels represent different times: the beginning of the simulation, the situation after 1 crossing time, the time of the bounce and the state after 2 crossing times.

tend to have all the same, small velocity. On the other hand, the resulting distribution when no gas is present tend to distribute the velocities of particles over a big range. This is the natural consequence of more efficient 2-body encounters which let the system segregate masses more rapidly.

## 6.4 Final considerations

The results presented in this chapter are just preliminary and a deeper analysis has still to be done. At present we evidenced the importance

of two dynamical mechanisms which contributes both to the resulting mass segregation of the system on very short time scales

1. the formation of substructures, as the system collapses, which segregate masses very quickly. This phenomenon acts *before* the bounce of the system because, during the bounce, the clumps are completely removed;

2. the formation of a very dense core which is responsible for the mass segregation of the system, *after* its bounce.

We also stressed the importance, on the resulting mass segregation, of the introduction of a stationary background potential (which mimics the presence of a gas) or of a heavier central object (for example a black hole) which both reduce the efficiency of close encounters between stars decreasing the efficiency of both energy exchanges and then of the resulting mass segregation. The analysis of density profiles, the inclusion of an expanding gas, the introduction of a more realistic mass spectrum and deeper considerations about the escaping stars in the different cases are surely very important points which will be taken into account in order to present final results on this work in a forthcoming publication. It is worth underlining that the simulations performed so far are very precise despite the criticality of the initial conditions of the tested systems (initially null velocities of the stars corresponding to the most violent collapse).

To confirm this point we show in Fig. 6.8 the relative variation of the total system energy for the case $N = 1,024$, $Q = 0$, no gas and the inclusion of the central heavier particle. It can be seen that the relative variation is always kept below the value of $10^{-4}$ even after the core collapse of the system, that occurs at time $t_{cc} \sim 7t_c$, after which very tight, central, multiple systems of stars form yielding to a deep increase of the total energy of the system.

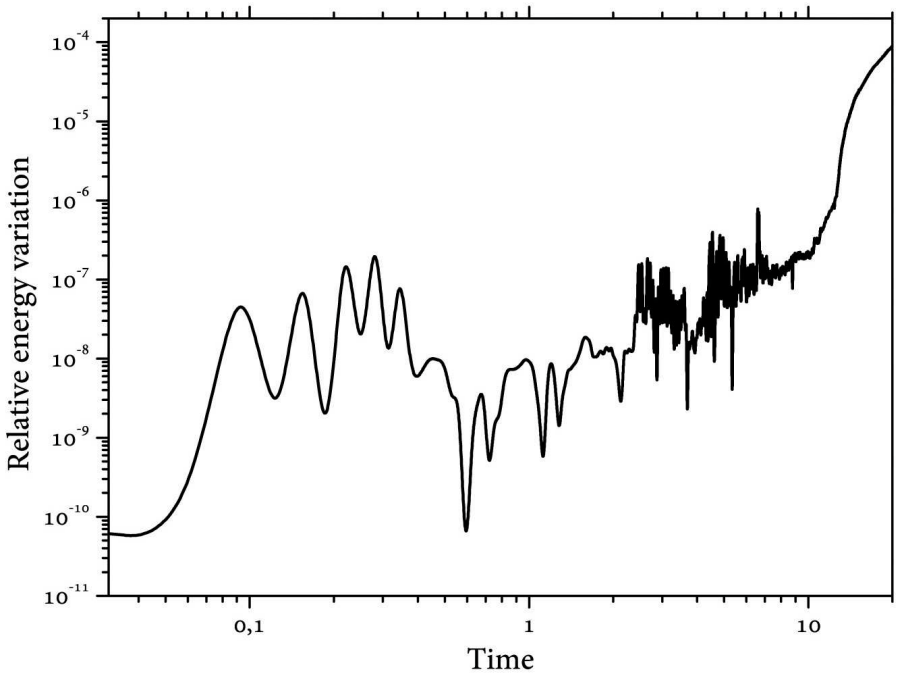


**Figure 6.8:** The relative energy variation for the most critical case simulated using our $N$-body code `HiGPUs` : $N = 1,024$, $Q = 0$ and the inclusion of a central stellar mass black hole. No gas (stationary background potential) is present.

Another point which must be stressed concerns some observations about the relaxation time. Spitzer [87] showed that the time needed by a system to segregate a population of stars with masses around $M$ is

$$t_{seg}\,(M) \simeq \frac{\langle m \rangle}{M} t_{rel} \tag{6.12}$$

where $\langle m \rangle$ is the average mass of the stars in the considered system[2]. Nevertheless, if the system is far from equilibrium, as it is in our cases, further simplifications of the formula for the relaxation time 6.2 are not

[2]This time has been calculated using the hypothesis of a spherical cluster containing a bimodal mass distribution with $m_H \gg m_L$ and total mass of heavy stars much less than the total mass of the core of the light stars

allowed. For example the assumptions of virial equilibrium and constant density simplify the formula 6.2 to the well known expression

$$t_{rel} \simeq \frac{N}{8 \log N} \frac{R}{\sigma}. \tag{6.13}$$

A less approximated expression, which does not assume a state of equilibrium or constant density, is the following

$$t_{rel} \simeq 0.34 \frac{\sigma^3}{G^2 m \rho \log \Lambda}. \tag{6.14}$$

In our simulations of the case $N = 1,024$, $Q = 0.0$, no gas and no black hole, the parameters of the dense core are $N \simeq 350$ stars, $R \simeq 15$ parsecs, $\sigma \simeq 3km/s$, $\langle m \rangle \simeq 1M_{\odot}$. If we substitute these parameters into the formula 6.13 we get 0.3 Myr which means that the system evolves fast and, on very short time scales (this very dense core lives, in our simulations, for about 0.2 Myr) can segregate masses up to $1.5M_{\odot}$. On the other hand, using the less approximated formula 6.14 we get that the relaxation time for this system is about 3 Myr therefore we have a system that can segregate masses down to $15M_{\odot}$. This is just one simple proof of the inconsistency and inapplicability of both formulas 6.13 and 6.14 to a situation which is very far from virial equilibrium.

# Acknowledgments

Questa è la prima volta che scrivo dei ringraziamenti a conclusione di un lavoro. Ho ritenuto opportuno farlo solo questa volta perché ci tengo a ringraziare esplicitamente alcune persone che hanno contribuito attivamente alla mia felicità e serenità durante questi $\sim 8$ anni di studi: una parte del mio sentimento di realizzazione personale è anche dovuto a loro. Comincio con una premessa: molto spesso le persone rimangono stupite dal fatto che io sia arrivato alla soglia di un dottorato di ricerca relativo ad una disciplina così "complicata" chiedendomi come io ci sia riuscito. Sperimentalmente, la velocità con cui si arriva alla domanda successiva è direttamente proporzionale alla retoricità della stessa; infatti essa riguarda, con grande probabilità, l'utilità dell'Astronomia o dell'Astrofisica. Sfortunatamente, molto spesso, chi fa questa domanda non la fa perché è conscio di non sapere e ha intenzione di allargare le proprie vedute ma, piuttosto, solo perché, con aria di sufficienza degna della peggior specie, ritiene la mia disciplina, a priori, decisamente inutile se paragonata a tutte le altre cose e, in special modo, alle loro occupazioni in ambito giuridico, medico, economico, ingegneristico etc etc .... Nei casi peggiori, *mentre* cerco di fornire risposte ai punti precedenti, si arriva alla fatidica questione universale: "Si ma... esistono gli alieni" la quale, ultimamente, si è unita a "Beh, insomma? Quando finisce il mondo?" (raccontandomi anche fantasiose esperienze personali annesse). Sembra proprio un quadro pessimo da come l'ho descritto ma, mentre qualche tempo fa cercavo di fornire risposte "esaurienti" ,

da un pò di tempo, a seconda della situazione, tento di divertirmi anche io fornendo delle risposte fantasiose la cui apocalitticità è inversamente proporzionale al grado di interesse mostrato dal "pubblico pagante" , proprio come si fa, d'altronde, in uno spettacolo circense. Dato questo quadro, si può immaginare quanto io sia grato a *Mamma* e *Papà* che ringrazio proprio per non far parte delle categorie elencate sopra e che, quindi, mi hanno permesso di andare a vivere a Roma (ancora diciottenne) e di non avermi mai impedito di inseguire un sogno che è partito da quando avevo circa 6 anni quando mi fissavo per ore a guardare il sistema solare rappresentato in un vecchio atlante di Papà. Ricordo benissimo il mio primo telescopio: il "famoso" Newton 114/900, nero e anche, come se fosse ieri, la mia felicità e le sensazioni provate quando, tornando a piedi da un rientro scolastico (era un Martedì di Dicembre), a soli 11 anni, ho visto i miei genitori che stavano lì, sul terrazzo, cercando di montare i pezzi. Ricordo bene che, la mia più grande paura, era di andare a dormire e che facesse giorno. Grazie ancora Mamma e grazie ancora Papà per avermi aiutato a maturare la mia passione fino al Dottorato di ricerca, cioè fino al grado più alto di istruzione e nel frattempo di non avermi mai trascurato; vi voglio bene e ve ne vorrò sempre.

Chiaramente però, nemmeno loro avrebbero potuto immaginare cosa sarebbe accaduto una volta entrato nel mondo dell'Università; infatti, a 18 anni, sono stato catapultato dalla vita tranquilla a casa dei miei genitori ad Avezzano, a Roma, in una casa dove non conoscevo nessuno (in casa c'eravamo io ed altri tre individui), da solo e senza particolari punti di riferimento. Qui è subentrata la fortuna, cioè quella di aver trovato subito (cosa che non accade in molti altri ambienti universitari) delle persone magnifiche. Sembra ieri che, dopo aver sbagliato strada per arrivare in Università, entrando per la prima volta in dipartimento (Nuovo Edifico di Fisica per l'esattezza) per il test di autovalutazione preliminare ai pre-corsi, il 7 Settembre 2005 alle ore 9:00 c.a., davanti l'aula 1 chiesi alla prima persona che incrociata: "Scusa, è qui che si fa

il test?" . Il caso vuole che quella persona, ancora oggi, può essere considerata, a tutti gli effetti, l'amico più importante, Filippo e subito dopo ne è subentrato un altro, Alessandro. Ancora mi ricordo il primo (di una lunghissima serie) laboratorio di fisica fatto insieme, le lunghe relazioni, le esperienze al telescopio fatte in montagna, le feste. . . , ma quanto ci siamo divertiti con quei cilindretti? Con loro ho passato i più bei momenti Universitari e, sebbene ormai io non riesca più a vedere spesso Alessandro (e forse, fra un pò, neanche Filippo) questa è una delle ultime occasioni per dirvi grazie e che vi voglio bene, moltissimo.

Non avrei mai immaginato che, dopo aver conosciuto due persone come Filippo e Alessandro, gli amici che tutti dovrebbero avere, le cose sarebbero andate ancora meglio. Infatti, durante il mio secondo anno di Università, ho conosciuto una dolce fanciulla, dall'aria inizialmente un pò indifferente ma a tratti radiosa come poche cose al mondo, che, di lì a qualche, mese sarebbe diventata la mia ragazza, Silvia. Non credo ci siano parole per descrivere quanto importante lei sia per me anche se lei sa benissimo che non c'è cosa al mondo che funzioni più di me e lei insieme. Non credevo si potesse raggiungere tale livello di intesa con una persona e tutt'oggi mi fermo spesso a pensare quanto io sia stato fortunato ad incontrarla. La serenità che mi dà, specie nei momenti peggiori, non è paragonabile a nessun'altra cosa, e lei è l'unica persona al mondo che è *sempre* riuscita a farmi sorridere. Da quel 17 Gennaio 2007 non c'è stato un litigio, un'incomprensione, un dubbio, nulla. Posso sicuramente dire, infatti, che senza Silvia non avrei avuto MAI un percorso scientifico così brillante quindi grazie infinite, anche se lei questo già lo sa perchè, comunque, cerco sempre di farle capire ogni giorno quanto sia importante per me e così continuerò sempre. Grazie anche a tutti i membri della sua famiglia (dai 0 ai $\sim$90 anni) per avermi sempre, accolto, ospitato e subito voluto bene.

In questi tre anni di Dottorato sicuramente una delle persone che va aggiunta esplicitamente alla lista è il mio collega (ma prima di tutto am-

ico) Manuel. Abbiamo felicemente condiviso la dependance della stanza 314 come dottorandi e abbiamo avuto numerose discussioni scientifiche, grazie alle quali siamo cresciuti molto entrambi. La sua presenza mi ha sempre messo di buon umore specialmente nell'andare in stanza 314, le mattine, perché pensavo: "Dai che pure oggi con Manuel tra una chiacchierata, una pausa, un'esperienza nuova e una risata portiamo a termine con successo, insieme, la giornata" . Lui ha contribuito a creare intorno a me un clima rilassato e sereno specie da quando, da un pò di mesi, condividiamo lo stesso appartamento a Roma. Sebbene, forse, in futuro prenderemo strade diverse sono sicuro che rimarremo per sempre legati. Nel frattempo ti ringrazio di essermi amico e di avermi fatto vivere un'esperienza di dottorato più unica che rara da tutti i punti di vista. Un altro posto di rilievo e quindi un sentito grazie, ovviamente, spetta anche al gruppo più stretto di amici di cui fanno parte sicuramente Grazia, Jacopo, Martina e Mauro: tutti insieme abbiamo passato dei momenti stupendi. Secondo me dobbiamo ritenerci molto fortunati, dei gruppi così, in genere, si formano solo nei licei e non in ambito universitario. Noi siamo un caso unico quindi impegniamoci, per favore, a non perderci mai. Non credo sia difficile considerata l'intesa che ci unisce e, credo che quando delle persone si sentono bene insieme, e hanno raggiunto la nostra maturità, sia sempre difficile separarle. Grazie a voi cari amici per aver condiviso con me una gran parte delle vostre esperienze in questi anni. Grazie sicuramente anche a Davide per l'impegno che ha dimostrato durante la sua laurea e per il primo anno di dottorato passato a lavorare insieme al fine di produrre il nostro primo articolo pubblicato su una rivista internazionale. Grazie soprattutto perché siamo poi diventati ottimi amici in breve tempo e sono sicuro che, sebbene lontani, il legame rimarrà sempre tale.

Grazie anche al Prof. Roberto Capuzzo Dolcetta, conosciuto durante il mio secondo anno di Università e che, da allora, mi ha seguito nella mia tesi triennale, specialistica e nel dottorato di ricerca. Egli continua ad essere il mio principale punto di riferimento scientifico da ormai più

di 6 anni e sicuramente a lui devo gran parte della mia crescita. Sicuramente devo ringraziarlo anche per avermi introdotto agli importanti personaggi del mio campo specifico (nominati più volte nella tesi) tra cui Sverre Aarseth, Seppo Mikkola, Rainer Spurzem, Peter Berczik e Keigo Nitadori. In particolare ringrazio Sverre, Seppo e Rainer per avermi ospitato e trattato come un loro pari rispettivamente presso l'Università di Cambridge, presso la cittadina di Turku e presso il National Astronomical Observatory of China a Pechino.

Merita un posto di rilievo anche Antonio, l'unica persona che è rimasta affianco a me dal liceo. Abbiamo passato (e continuiamo a trascorrere) tante belle serate insieme e rimane ancora oggi la persona con cui ho passeggiato di più in giro per Roma ammirandone le bellezze (specie durante i primi anni di Università). Grazie anche a te, caro Tony, per la compagnia che mi fai partendo e tornando da/ad Avezzano in macchina. Sicuramente hai contribuito moltissimo alla mia serenità e felicità, quindi parte del merito della realizzazione di questo obiettivo è anche tuo.

Infine grazie infinite alla mia "compagna d'inglese" Claudia che mi ha aiutato a migliorare il mio inglese (a dir la verità me lo ha proprio insegnato). La conosco ormai da più di 3 anni e, senza di lei, non sarei mai potuto intervenire in conferenze internazionali e, di certo, non avrei mai potuto scrivere questa tesi.

Grazie ancora alle persone menzionate che mi hanno sempre reso sereno e felice creando un clima di tranquillità che sta dietro alla mia realizzazione personale e, in modo particolare, a questa tesi di dottorato che, a grandi linee, conclude il mio percorso di studi universitari.

*Grazie a tutti*

Mario

# Bibliography


[1] S. J. Aarseth. *Direct n-body calculations*. Published in *Dynamics of Star Clusters*. Ed. by J. Goodman and P. Hut. Vol. 113. IAU Symposium. 1985. Pp. 251–258. (Cit. on p. 10).

[2] S. J. Aarseth. *Direct N-body codes*. Published in *The Cambridge N-Body Lectures*. Ed. by S. J. Aarseth, C. A. Tout, and R. A. Mardling. Springer-Verlag, Berlin Heidelberg, 2008. Pp. 1–30. (Cit. on p. 143).

[3] S. J. Aarseth. *From NBODY1 to NBODY6: The Growth of an Industry*. *pasp* 111 (Nov. 1999), pp. 1333–1346 (cit. on pp. 34, 68).

[4] S. J. Aarseth. *Gravitational N-Body Simulations : tools and algorithms*. The Edinburgh Building, Cambridge CB2 8RU, UK: Cambridge University Press, 2003 (cit. on pp. 47, 169, 173).

[5] S. J. Aarseth, J. P. Anosova, V. V. Orlov, and V. G. Szebehely. *Global chaoticity in the Pythagorean three-body problem*. *Celestial Mechanics and Dynamical Astronomy* 58 (Jan. 1994), pp. 1–16 (cit. on p. 190).

[6] S.J. Aarseth, D.N.C. Lin, and J.C.B. Papaloizou. *On the collapse and violent relaxation of protoglobular clusters*. *apj* 324 (Jan. 1988), pp. 288–310 (cit. on p. 193).

[7] A. Ahmad and L. Cohen. *A numerical integration scheme for the N-body gravitational problem*. *Journal of Computational Physics* 12 (1973), pp. 389–402 (cit. on p. 35).

[8] C. Allen and A. Santillan. *An improved model of the galactic mass distribution for orbit computations*. *rmxaa* 22 (Oct. 1991), pp. 255–263 (cit. on p. 101).

[9] R.J. Allison, S.P. Goodwin, R.J. Parker, et al. *Dynamical Mass Segregation on a Very Short Timescale*. *apj* 700 (Aug. 2009), pp. L99–L103 (cit. on p. 192).

[10] R.J. Allison, S.P. Goodwin, R.J. Parker, et al. *Using the minimum spanning tree to trace mass segregation*. *mnras* 395 (May 2009), pp. 1449–1454 (cit. on p. 197).

[12] F. Antonini, R. Capuzzo-Dolcetta, A. Mastrobuono-Battisti, and D. Merritt. *Dissipationless Formation and Evolution of the Milky Way Nuclear Star Cluster*. *ApJ* 750 (May 2012), pp. 111–125. arXiv: 1110.5937 [astro-ph.GA] (cit. on pp. 93, 117).

[13] J. Barnes and P. Hut. *A hierarchical O(N log N) force-calculation algorithm*. *nat* 324 (Dec. 1986), pp. 446–449 (cit. on p. 35).

[14] J. Bédorf and S. Portegies Zwart. *A pilgrimage to gravity on GPUs*. *European Physical Journal Special Topics* 210 (Aug. 2012), pp. 201–216. arXiv: 1204.3106 [astro-ph.IM] (cit. on p. 35).

[15] K. Bekki. *The Formation of Stellar Galactic Nuclei through Dissipative Gas Dynamics*. *PASA* 24 (July 2007), pp. 77–94 (cit. on p. 92).

[16] P. Berczik, R. Spurzem, I. Berentzen, et al. *High performance massively parallel direct N-body simulations on large GPU clusters*. *Proceedings of International conference on High Performance Computing 2011 Kyev, Ukraine* (2011), pp. 8–18 (cit. on pp. 34, 62, 68, 71, 86, 90).

[17] P. Berczik, K. Nitadori, T. Hamada, and R. Spurzem. *The parallel NBody code phiGPU*. *New Astronomy, in preparation* (2011) (cit. on p. 62).

[18] J. Binney and S. Tremaine. *Galactic Dynamics*. 41 William Street, Princeton, New Jersey 08540: Princeton University Press, 2008 (cit. on pp. 29, 126, 128, 129, 176, 192).

[19] D. Boccaletti and G. Pucacco. *Theory of Orbits: Volume 1: Integrable Systems and Non-perturbative Methods*. Springer-Verlag Berlin Heidelberg New York, 2003 (cit. on pp. 6, 15).

[20] I. A. Bonnell and M. B. Davies. *Mass segregation in young stellar clusters*. *MNRAS* 295 (1998), pp. 691–698 (cit. on p. 193).

[21] C. A. Burdet. *Regularization of the two-body problem*. *Z. Angew. Math. Phys.* 18 (1967), pp. 434–442 (cit. on p. 166).

[22] R. Capuzzo-Dolcetta. *The Evolution of the Globular Cluster System in a Triaxial Galaxy: Can a Galactic Nucleus Form by Globular Cluster Capture? ApJ* 415 (Oct. 1993), pp. 616–630. eprint: `arXiv:astro-ph/9301006` (cit. on p. 92).

[23] R. Capuzzo-Dolcetta and P. Miocchi. *A comparison between the fast multipole algorithm and the tree-code to evaluate gravitational forces in 3-D. Journal of Computational Physics* 143 (June 1998), pp. 29–48. eprint: arXiv:astro-ph/9703122 (cit. on p. 35).

[24] R. Capuzzo-Dolcetta and P. Miocchi. *Merging of Globular Clusters in Inner Galactic Regions. II. Nuclear Star Cluster Formation*. *ApJ* 681 (July 2008), pp. 1136–1147. arXiv: 0801.1072 (cit. on p. 93).

[25] R. Capuzzo-Dolcetta and M. Spera. *A performance comparison of different graphics processing units running direct N-body simulations*. *Computer Physics Communications* 184 (2013), pp. 2528–2539 (cit. on pp. 69, 94).

[26] R. Capuzzo-Dolcetta and M. Spera. *High Precision Simulations of the Evolution of a Super Star Cluster Around a Massive Black Hole*. Published in *Advances in Computational Astrophysics: Methods, Tools, and Outcomes*. Ed. by R. Capuzzo-Dolcetta, M. Limongi, and A. Tornambe'. Astronomical Society of the Pacific, 2012. Pp. 351–352. (Cit. on p. 86).

[27] R. Capuzzo-Dolcetta and L. Vignola. *Have many globulars disappeared to the galactic centres? The case of the Galaxy, M 31 and M 87*. *A&A* 327 (Nov. 1997), pp. 130–136 (cit. on p. 92).

[28] R. Capuzzo-Dolcetta, M. Spera, and D. Punzo. *A fully parallel, high precision, N-body code running on hybrid computing platforms*. *Journal of Computational Physics* 236 (Mar. 2013), pp. 580–593. arXiv: 1207.2367 [astro-ph.IM] (cit. on pp. 34, 64, 69, 83).

[29] R. Capuzzo-Dolcetta, A. Mastrobuono-Battisti, and D. Maschietti. *NBSymple, a double parallel, symplectic N-body code running on graphic processing units*. *na* 16 (July 2011), pp. 284–295. arXiv: 1003.3896 [astro-ph.IM] (cit. on pp. 34, 41).

[30] R. Capuzzo-Dolcetta, M. Arca-Sedda, and M. Spera. *The Dense Stellar Systems Around Galactic Massive Black Holes*. *ArXiv e-prints* (Feb. 2013). arXiv: 1302.2509 [astro-ph.CO] (cit. on p. 117).

[31] T. Carleman. *Ueber die Abelsche Integralgleichung mit konstanten Integrationsgrenzen*. *Math. Z.* 15 (1922), pp. 111–120 (cit. on p. 130).

[33] R. Clausius. *On a mechanical theorem applicable to heat*. *Phil. Mag. Ser. 4* XL (1870), pp. 122–127 (cit. on p. 19).

[38] W. Dehnen. *A Family of Potential-Density Pairs for Spherical Galaxies and Bulges*. *MNRAS* 265 (Nov. 1993), p. 250 (cit. on pp. 132, 150).

[39] W. Dehnen and J. I. Read. *N-body simulations of gravitational dynamics*. *European Physical Journal Plus* 126 (2011), p. 55 (cit. on p. 34).

[40] A. S. Eddington. *The distribution of stars in globular clusters*. *MNRAS* 76 (1916), pp. 572–585 (cit. on p. 131).

[41] E. Elsen, V. Vishal, M. Houston, et al. *N-Body Simulations on GPUs*. *CoRR* abs/0706.3060 (2007) (cit. on p. 89).

[42] R.T. Farouki, G.L. Hoffman, and E.E. Salpeter. *The collapse and violent relaxation of N-body systems - Mass segregation and the secondary maximum*. *apj* 271 (Aug. 1983), pp. 11–21 (cit. on p. 192).

[43] H. Flanders. *Differentiation Under the Integral Sign*. *The American Mathematical Monthly* 80, No.6 (1973), pp. 615–627 (cit. on p. 130).

[44] M. Fujii, M. Iwasawa, Y. Funato, and J. Makino. *BRIDGE: A Direct-Tree Hybrid N-Body Algorithm for Fully Self-Consistent Simulations of Star Clusters and Their Parent Galaxies*. *pasj* 59 (Dec. 2007), pp. 1095–. arXiv: 0706.2059 (cit. on p. 35).

[45] E. Gaburov, S. Harfst, and S. Portegies Zwart. *SAPPORO: A way to turn your graphics cards into a GRAPE-6*. *New Astronomy* 14 (2009), pp. 630–637 (cit. on pp. 62, 68).

[46] W. B. Gragg. *On extrapolation algorithms for ordinary initial value problems*. *SIAM J. Num. Anal.* 2 (1965), pp. 384–403 (cit. on p. 139).

[47] L. Greengard and V. Rokhlin. *A fast algorithm for particle simulations*. *Journal of Computational Physics* 73 (Dec. 1987), pp. 325–348 (cit. on p. 35).

[49] S. Harfst, A. Gualandris, D. Merritt, et al. *Performance analysis of direct N-body algorithms on special-purpose supercomputers*. 12 (July 2007), pp. 357–377. eprint: arXiv:astro-ph/0608125 (cit. on p. 34).

[50] P.H. Hauschildt and E. Baron. *A 3D radiative transfer framework. VIII. OpenCL implementation*. *aap* 533, A127 (Sept. 2011), A127 (cit. on p. 121).

[51] D.C. Heggie. *Regularization using a time-transformation only*. Published in *Recent advances in Dynamical Astronomy*. Ed. by Springer Netherlands. Vol. 39. Astrophysics and Space Science Library. 1973. Pp. 34–37. (Cit. on p. 166).

[52] L.A. Hillenbrand and L.W. Hartmann. *A Preliminary Study of the Orion Nebula Cluster Structure and Dynamics*. *apj* 492 (Jan. 1998), p. 540 (cit. on p. 191).

[53] R. W. Hockney and J. W. Eastwood. *Computer simulation using particles*. 270 Madison Avenue, New York, NY 10016: Taylor and Francis Group, 1988 (cit. on p. 35).

[54] E. Holmberg. *On the Clustering Tendencies among the Nebulae. II. a Study of Encounters Between Laboratory Models of Stellar Systems by a New Integration Procedure*. *ApJ* 94 (Nov. 1941), p. 385 (cit. on pp. 8, 9).

[55] J. H. Jeans. *On the theory of star-streaming and the structure of the universe*. *MNRAS* 76 (Dec. 1915), pp. 70–84 (cit. on p. 127).

[56] I.R. King. *The structure of star clusters. III. Some simple dynamical models*. *Astrophysical Journal* 71 (1966), p. 64 (cit. on pp. 101, 103, 133, 147).

[57] H. Kirk, D. Johnstone, and M. Tafalla. *Dynamics of Dense Cores in the Perseus Molecular Cloud*. *apj* 668 (Oct. 2007), pp. 1042–1063 (cit. on p. 195).

[58] S. Konstantinidis and K. D. Kokkotas. *MYRIAD: a new N-body code for simulations of star clusters*. *aap* 522, A70 (Nov. 2010), A70. arXiv: `1006.3326 [astro-ph.IM]` (cit. on p. 34).

[59] P. Kroupa. *On the variation of the initial mass function*. *mnras* 322 (Apr. 2001), pp. 231–246. eprint: `arXiv:astro-ph/0009005` (cit. on p. 101).

[60] A. H. W. Kupper, T. Maschberger, P. Kroupa, and H. Baumgardt. *Mass segregation and fractal substructure in young massive clusters - I. The McLuster code and method calibration*. *MNRAS* 417 (Nov. 2011), pp. 2300–2317. arXiv: 1107.2395 (cit. on pp. 101, 138).

[61] P. Kustaanheimo and E. Stiefel. *Perturbation theory of Kepler motion based on spinor regularization*. *Journal für die Reine und Angewandte Mathematik* 218 (1965), pp. 204–219 (cit. on p. 169).

[62] T. Levi-Civita. *Sur la régularisation du problème des trois corps*. *Acta Mathematica* 42 (1920), pp. 99–144 (cit. on p. 169).

[63] D. Li, J. Kauffmann, Q. Zhang, and W. Chen. *Massive Quiescent Cores in Orion: Dynamical State Revealed by High-resolution Ammonia Maps*. *apjl* 768 (May 2013), p. L5 (cit. on p. 195).

[64] J. Makino. *A modified Aarseth code for GRAPE and vector processors*. *Publ. Astron. Soc. Japan* 43 (1991), pp. 859–76 (cit. on p. 43).

[65] J. Makino. *Optimal order and time-step criterion for Aarseth-type N-body integrators*. *Astrophys. J.* 369 (1991), pp. 200–12 (cit. on p. 43).

[66] Makoto Matsumoto and Takuji Nishimura. *Mersenne Twister: A 623-dimensionally Equidistributed Uniform Pseudo-random Number Generator*. *ACM Trans. Model. Comput. Simul.* 8.1 (Jan. 1998), pp. 3–30 (cit. on p. 198).

[67] S. McMillan, E. Vesperini, and N. Kruczek. *Rapid Mass Segregation in Massive Star Clusters*. *ArXiv e-prints* (Oct. 2012). arXiv: 1210.8200 (cit. on p. 193).

[68] S. McMillan, S. Portegies Zwart, A. van Elteren, and A. Whitehead. *Simulations of Dense Stellar Systems with the AMUSE Software Toolkit*. Published in *Advances in Computational Astrophysics: Methods, Tools, and Outcome*. Ed. by R. Capuzzo-Dolcetta, M. Limongi, and A. Tornambè. Vol. 453. Astronomical Society of the Pacific Conference Series. July 2012. P. 129. arXiv: `1111.3987 [astro-ph.IM]`. (Cit. on p. 92).

[69] S. L. W. McMillan. *The vectorization of small-N integrators*. Published in *The Use of Supercomputers in Stellar Dynamics*. Ed. by P. Hut and S. McMillan. Vol. 267. Springer-Verlag, Berlin Heidelberg New York, 1986. P. 156. (Cit. on p. 44).

[70] S.L.W. McMillan, E. Vesperini, and S.F. Portegies Zwart. *A Dynamical Origin for Early Mass Segregation in Young Star Clusters*. *apjl* 655 (Jan. 2007), pp. L45–L49 (cit. on p. 193).

[71] S. Mikkola. *Numerical Treatment of Small Stellar Systems with Binaries*. *Celestial Mechanics and Dynamical Astronomy* 68 (May 1997), pp. 87–104 (cit. on p. 174).

[72] S. Mikkola and S. Aarseth. *An implementation of N-body chain regularization*. *Cel. Mech. Dyn. Ast.* 57 (1993), p. 439 (cit. on p. 172).

[73] S. Mikkola and K. Tanikawa. *Explicit symplectic algorithms for time-transformed Hamiltonians*. *Celest. Mech. Dyn. Astr.* 74 (1999), pp. 287–295 (cit. on p. 174).

[74] M. Milosavljević. *On the Origin of Nuclear Star Clusters in Late-Type Spiral Galaxies*. *ApJL* 605 (Apr. 2004), pp. L13–L16. eprint: `arXiv:astro-ph/0310574` (cit. on p. 92).

[75] P. Miocchi and R. Capuzzo-Dolcetta. *An efficient parallel tree-code for the simulation of self-gravitating systems*. *aap* 382 (Feb. 2002), pp. 758–767. eprint: arXiv:astro-ph/0104152 (cit. on p. 35).

[76] A. Munshi, B. R. Gaster, T. G. Mattson, J. Fung, and D. Ginsburg. *OpenCL Programming Guide*. Addison-Wesley, 2012 (cit. on p. 59).

[77] K. Nitadori and S. J. Aarseth. *Accelerating NBODY6 with graphics processing units*. *mnras* 424 (July 2012), pp. 545–552. arXiv: 1205.1222 [astro-ph.IM] (cit. on pp. 35, 62).

[78] K. Nitadori and J. Makino. *Sixth- and eighth-order Hermite integrator for N-body simulations*. *New Astronomy* 13 (2008), pp. 498–507 (cit. on pp. 48, 50, 72, 90).

[79] L. Nyland, M. Harris, and J. Prins. *Fast N-Body Simulation with CUDA*. Vol. 31. Prentice-Hall, 2007. Chap. 31, pp. 677–695 (cit. on pp. 63, 89).

[82] H. C. Plummer. *On the problem of distribution in globular star clusters*. *mnras* 71 (1911), pp. 460–470 (cit. on pp. 24, 33, 70, 99, 131).

[83] S. F. Portegies Zwart, S. L. W. McMillan, P. Hut, and J. Makino. *Star cluster ecology - IV. Dissection of an open star cluster: photometry*. *mnras* 321 (Feb. 2001), pp. 199–226. eprint: arXiv:astro-ph/0005248 (cit. on p. 34).

[84] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge University Press, 2007 (cit. on pp. 139, 140, 186).

[85] M. Preto and S. Tremaine. *A Class of Symplectic Integrators with Adaptive Time Step for Separable Hamiltonian Systems*. *aj* 118 (Nov. 1999), pp. 2532–2541. eprint: `astro-ph/9906322` (cit. on p. 174).

[86] J. Sanders and E. Kandrot. *CUDA by example: an Introduction to General-Purpose GPU Programming*. Addison-Wesley, 2010 (cit. on p. 59).

[87] L. Spitzer Jr. *Equipartition and the Formation of Compact Nuclei in Spherical Stellar Systems*. *apj* 158 (Dec. 1969), p. L139 (cit. on p. 209).

[88] V. Springel. *The cosmological simulation code GADGET-2*. *mnras* 364 (Dec. 2005), pp. 1105–1134. eprint: `arXiv:astro-ph/0505010` (cit. on p. 36).

[89] R. Spurzem, P. Berczik, I. Berentzen, et al. *Astrophysical particle simulations with large custom GPU clusters on three continents*. *Computer Science - Research and Development* 26 (3 2011), pp. 145–151 (cit. on p. 90).

[90] V. Szebehely and C.F. Peters. *A new periodic solution of the problem of three bodies*. *aj* 72 (Nov. 1967), p. 1187 (cit. on p. 190).

[92] P. Teuben. *The Stellar Dynamics Toolbox NEMO*. Published in *Astronomical Data Analysis Software and Systems IV*. Ed. by R. A. Shaw, H. E. Payne, and J. J. E. Hayes. Vol. 77. Astronomical Society of the Pacific Conference Series. 1995. P. 398. (Cit. on p. 138).

[96] Q.D. Wang. *The global solution of the n-body problem*. *Celest. Mech. Dyn. Astr.* 50 (1991), pp. 73–88 (cit. on pp. 5, 7).

[97] N. Wilt. *The CUDA Handbook*. Addison-Wesley, 2013 (cit. on p. 59).

[98] H. Yoshida. *Construction of higher order symplectic intergators*. *Phys. Lett. A* 150 (1990), p. 262 (cit. on p. 41).

## Web sites

[11] Advanced Micro Devices (AMD). *OpenCL Zone*. URL: http://developer.amd.com/resources/heterogeneous-computing/opencl-zone/ (cit. on p. 59).

[32] *CINECA Web Page*. URL: http://www.cineca.it/en (cit. on p. 69).

[34] nVIDIA corporation. *CUDA C programming guide*. 2013. URL: http://docs.nvidia.com/cuda/pdf/CUDA_C_Programming_Guide.pdf (cit. on pp. 58, 59, 89).

[35] nVIDIA corporation. *CUDA Zone*. URL: https://developer.nvidia.com/category/zone/cuda-zone (cit. on p. 59).

[36] nVIDIA corporation. *NVIDIA's Next Generation CUDA Compute Architecture: Fermi*. 2009. URL: http://www.nvidia.com/content/PDF/fermi_white_papers/NVIDIA_Fermi_Compute_Architecture_Whitepaper.pdf (cit. on pp. 54, 59, 89).

[37] nVIDIA corporation. *NVIDIA's Next Generation CUDA Compute Architecture: Kepler GK110*. 2013. URL: http://www.nvidia.com/content/PDF/kepler/NVIDIA-Kepler-GK110-Architecture-Whitepaper.pdf (cit. on pp. 54, 59).

[48] Khronos Group. *OpenCL Web Page*. URL: http://www.khronos.org/opencl/ (cit. on p. 59).

[80] *Open MPI Web Page*. URL: http://www.open-mpi.org/ (cit. on p. 64).

[81] *OpenCL Web Page*. URL: http://openmp.org/wp/ (cit. on p. 64).

[91] Alan Tatourian. *nVIDIA GPU architecture and CUDA programming environment*. URL: http://tatourian.com/2013/09/03/nvidia-gpu-architecture-cuda-programming-environment/ (cit. on p. 53).

[93] *The TH-1A Supercomputer*. URL: http://www.nscc-tj.gov.cn/en/resources/resources_1.asp (cit. on p. 61).

[94] *The Titan supercomputer*. URL: http://www.olcf.ornl.gov/titan/ (cit. on pp. 61, 117).

[95] *Top 500 list*. URL: http://www.top500.org/ (cit. on pp. 61, 91).